Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту»

Виконала: студентка 4 курсу
групи 2ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Волос А.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Мельник О.В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Богомолов С.В.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Волос Анастасії Вікторівні

1. Тема роботи – «Розробка вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту»

Керівник роботи: Мельник Олександр Васильович, к. т. н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студенткою роботи 2 червня 2025 р.

3. Вихідні дані до роботи: тип програми – вебзастосунок; модель розробки – інкрементна; засоби організації даних програми – бібліотеки React, Node.js; вхідні дані: ім'я користувача, вага, цільові показники калорій та ваги, дані про час сну; вихідні дані: інтерфейс користувача, графіки зі статистикою, аналіз спожитих страв; середовище розробки – Visual Studio Code; мова програмування – Typescript.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку вебсервісів для контролю біологічних показників людини з використанням штучного інтелекту; розробка структури та алгоритмів програмного застосунку; розробка системи для моніторингу біологічних показників людини і підтримки способу життя; тестування програмного забезпечення; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; ме
об'єкт та предмет дослідження; задачі дослідження, наукова новизна, практи
цінність одержаних результатів; порівняльний аналіз аналогів, використ
технології; розробка методу автоматичного аналізу харчування на осн
комп'ютерного зору; розробка методу персоналізованих адаптивних рекомендац
тестування; висновки; апробація результатів роботи та публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Мельник О.В., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку вебсервісів для контролю біологічних показників людини з використанням штучного інтелекту	25.03.25 – 06.04.25	вик
2	Розробка структури та алгоритмів програмного застосунку	07.04.25 – 20.04.25	вик
3	Варіантний аналіз та вибір мови програмування розробки	20.04.25 – 07.05.25	вик
4	Розробка системи для моніторингу біологічних показників людини і підтримки способу життя	07.05.25 – 20.05.25	вик
5	Тестування програмного забезпечення	20.05.25– 30.05.25	вик
6	Оформлення матеріалів до захисту БКР	25.03.25 – 06.04.25	вик

Студентка _____ (підпис) Волос А.В. (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) Мельник О.В. (прізвище та ініціали)

АНОТАЦІЯ

УДК 004:005.9

Волос А.В. Розробка вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту: бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця : ВНТУ, 2025. 95 с.

У бакалаврській кваліфікаційній роботі було розроблено вебсервіс для контролю біологічних показників людини з використанням штучного інтелекту. Розроблений сервіс дозволяє відстежувати зміну ваги користувача, проводити аналіз спожитої їжі, випитої води, отримувати рекомендації щодо тренування та харчування.

У ході роботи було проведено аналіз інформаційного забезпечення вебсервісу, визначено дані, які збираються, обробляються, генеруються при роботі вебсервісу. Розроблено модель системи у три шари – клієнтська частина, серверна та мережева. Описано компоненти, що використовуються у роботі вебсервісу. Розроблено архітектуру вебсервісу. Розроблено методи автоматичного аналізу харчування на основі комп’ютерного зору та персоналізованих адаптивних рекомендацій.

Програмне забезпечення розроблено із використанням середовища розробки Visual Studio Code з використанням мови програмування Typescript. У процесі розробки використовувались сучасні бібліотеки та інструменти, такі як React, Node.js, Claude API. Проведено тестування розробленого сервісу. Було продемонстровано його можливості, наведено приклад використання, продемонстровано, що розроблений застосунок виконує поставлені на початку розробки завдання.

Ключові слова: штучний інтелект, вебсервіс, біологічні показники.

ABSTRACT

UDC 004:005.9

Volos A.V. Development of a web service for monitoring and optimizing human biological indicators using artificial intelligence: bachelor's qualification work in specialty 121 Software Engineering, educational program – Software Engineering. Vinnytsia: VNTU, 2025. 95 p.

In the bachelor's qualification work, a web service was developed for monitoring and optimizing human biological indicators using artificial intelligence. The developed service allows you to track changes in the user's weight, analyze food consumed, water drunk, and receive recommendations for training and nutrition.

During the work, an analysis of the information support of the web service was conducted, the data that is collected, processed, and generated during the operation of the web service was determined. A system model was developed in three layers – client part, server part, and network part. The components used in the operation of the web service were described. The architecture of the web service was developed. Methods for automatic nutrition analysis based on computer vision and personalized adaptive recommendations were developed.

The software was developed using the Visual Studio Code development environment using the Typescript programming language. Modern libraries and tools such as React, Node.js, Claude API were used in the development process. The developed service was tested. Its capabilities were demonstrated, an example of use was given, and it was demonstrated that the developed application performs the tasks set at the beginning of development.

Keywords: artificial intelligence, web service, biological indicators.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Моніторинг здоров‘я, або моніторинг mHealth, популярний у цифровому секторі охорони здоров’я завдяки використанню портативних пристроїв, таких як смартфони, планшети та портативні комп‘ютери, для збору та передачі інформації про здоров‘я з метою керування здоров‘ям користувача [1].

Прагнення сучасної людини до здорового способу життя призвело до популяризації додатків для дослідження біологічних показників. Вони полегшують відстеження калорій, активності, сну та інших важливих показників здоров‘я. Однак ці застосунки характеризуються неточністю даних, бездіяльністю користувачів та інформаційною безпекою. У контексті технологічного прогресу вдосконалення цих ініціатив є пріоритетом для підвищення їх ефективності та цінності. Трекери здоров'я пропонують користувачам інструменти для спостереження за своїм фізичним здоров'ям на особистому рівні [2].

Багато додатків пропонують неперсоналізовані рекомендації, що не враховують індивідуальних потреб та стану здоров'я користувача. Це може призводити до низької ефективності програм або навіть до шкоди.

Перевантаження даними без зрозумілих візуалізацій ускладнює інтерпретацію інформації та її корисність. Проблеми з мотивацією і тривалістю використання пов'язані з недостатньою гейміфікацією та відсутністю індивідуальних досягнень. Надійність і точність даних є важливими аспектами, оскільки неточні дані можуть негативно вплинути на прийняття рішень користувачами. Останнім, але не менш важливим аспектом є забезпечення конфіденційності та безпеки особистих даних, що є важливим для довіри користувачів до таких додатків.

Отже, актуальною є розробка застосунку для моніторингу біологічних показників людини.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі

програмного забезпечення.

Метою роботи є підвищення рівня контролю біологічних показників людини шляхом розробки спеціалізованої системи, що дозволяє відслідковування та контроль біологічних показників людини.

Основними задачами дослідження є:

– розробити архітектуру застосунку для контролю біологічних показників людини;

– розробити інтерфейс користувача застосунку;

– розробити алгоритми роботи застосунку;

– розробити функціонал застосунку;

– провести тестування розробленого застосунку.

Об'єкт дослідження – процес розробки системи для контролю біологічних показників людини з використанням штучного інтелекту.

Предмет дослідження – методи та засоби розробки системи для контролю біологічних показників людини з використанням штучного інтелекту.

Методи дослідження. У процесі досліджень використовувались: емпіричні методи розробки вебзастосунків, методи об’єктно-орієнтованого програмування для розробки модулів вебсервісу, методи комп’ютерного зору для розпізавання спожитих продуктів харчування, методи штучного інтелекту для аналізу харчування, методи тестування вебсервісів.

Новизна отриманих результатів:

1. Подальшого розвитку отримав метод автоматичного аналізу харчування на основі комп’ютерного зору, який, на відміну від відомих, дозволяє провести аналіз нутрієнтної цінності раціону харчування по його фото.

2. Подальшого розвитку отримав метод персоналізованих адаптивних рекомендацій, який на відміну від відомих, надає персоналізовані рекомендації, які залежать від харчування користувача, його фізичної активності, бажаної ваги та мети використання застосунку.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає у можливості використання розробленої системи

для контролю біологічних показників людини, аналізу харчування, відслідковування ваги.

Особистий внесок здобувача. Усі результати, викладені в бакалаврській кваліфікаційній роботі, отримані автором особисто. У роботі, виконаній у співавторстві, автору належить: аналіз стану застосунків для моніторингу здоров'я, перелік вирішених завдань.

Апробація результатів роботи. Описані у роботі положення доповідались на конференції «IV Всеукраїнська науково-технічна конференції молодих вчених, аспірантів і студентів».

Публікації. Результати роботи були опубліковані в матеріалах конференції «IV Всеукраїнська науково-технічна конференції молодих вчених, аспірантів і студентів» [3].

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто 4 розділи та було використано 25 літературних джерел.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБСЕРВІСІВ ДЛЯ КОНТРОЛЮ БІОЛОГІЧНИХ ПОКАЗНИКІВ ЛЮДИНИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Аналіз стану питання розробки

Розробка вебсервісів для контролю біологічних показників людини із застосуванням технологій штучного інтелекту знаходиться на етапі динамічного розвитку та трансформації. Останні роки характеризуються значним прогресом у створенні інтелектуальних систем моніторингу здоров'я та харчування, що поєднують аналітичні можливості штучного інтелекту з доступністю вебтехнологій.

Сучасний ринок представлений кількома категоріями застосунків: трекери харчування з функціями розпізнавання їжі [4], комплексні системи моніторингу здоров'я, персоналізовані рекомендаційні платформи та інтегровані екосистеми здоров'я. Необхідно відзначити, що більшість існуючих сервісів фокусуються на окремих аспектах здоров'я, пропонуючи обмежену інтеграцію різних біологічних показників [5].

В контексті технологічних інновацій спостерігається виразний тренд до впровадження методів комп'ютерного зору для аналізу харчування [6]. Однак більшість існуючих систем базуються на спеціалізованих моделях машинного навчання.

Паралельно з цим розвивається напрямок персоналізованих рекомендаційних систем у сфері здоров'я. Дослідження Harvard T.H. Chan School of Public Health демонструють, що персоналізовані нутритивні рекомендації, адаптовані до індивідуальних біологічних показників, показують ефективність на 37 % вищу порівняно з загальними рекомендаціями [7].

Незважаючи на технологічний прогрес, галузь стикається з рядом суттєвих викликів. Питання точності розпізнавання їжі залишається актуальним, особливо для складних багатокомпонентних страв або регіональних кулінарних традицій. Згідно з дослідженням Journal of Medical Internet Research, навіть провідні системи

розпізнавання їжі демонструють точність не вище 85 % у реальних умовах використання.

Сучасний тренд інтеграції різних типів біологічних даних (харчування, фізична активність, сон, стрес, біохімічні показники) вимагає розробки складних архітектур даних та алгоритмів аналізу [8]. Провідні гравці ринку створюють екосистеми для агрегації різнорідних показників здоров’я, однак комплексний аналіз цих даних залишається обмеженим.

Ринок інтелектуальних систем для контролю здоров’я демонструє стійке зростання. Аналіз користувацьких потреб виявляє високий попит на інтегровані рішення з простим інтерфейсом. Згідно з опитуванням JMIR mHealth, 78 % користувачів припиняють використання додатків для відстеження здоров’я протягом перших 30 днів через складність введення даних [9]. Тому актуальною є розробка вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту.

1.2 Порівняльний аналіз аналогів

Samsung Health [10] – це універсальний додаток для моніторингу здоров’я, розроблений компанією Samsung, який об'єднує численні інструменти для відстеження фізичної активності, сну, харчування та інших життєвих показників. Додаток створено для користувачів, які прагнуть вести збалансований спосіб життя та отримувати персоналізовані рекомендації для покращення здоров’я. Він підтримує інтеграцію з різноманітними пристроями, включаючи смартфони, фітнес-трекери та інші носимі гаджети.

Функціонал Samsung Health включає моніторинг фізичної активності за допомогою підрахунку кроків, дистанції та спалених калорій, а також відстеження сну, рівня стресу та пульсу. Додаток дозволяє користувачам реєструвати прийоми їжі, контролювати споживання калорій та води, що допомагає в управлінні вагою та харчуванням. Інтерактивні графіки та аналітичні звіти надають змогу візуалізувати прогрес та коригувати режим дня.

Інтерфейс Samsung Health наведено на рисунку 1.1.

Рисунок 1.1 – Інтерфейс Samsung Health

MyFitnessPal [11] є одним із найпопулярніших додатків для моніторингу харчування та контролю ваги, що отримав широке визнання завдяки своїй великій базі даних продуктів. Додаток дозволяє легко реєструвати прийоми їжі, завдяки функції сканування штрих-кодів, що спрощує процес введення даних.

Основні функції MyFitnessPal включають: підрахунок калорій, моніторинг

макро- та мікронутрієнтів, а також відстеження фізичної активності. Завдяки інтеграції з різноманітними фітнес-додатками та носимими пристроями, сервіс дозволяє отримувати комплексну картину про енергетичний баланс користувача. Додаток також дозволяє встановлювати індивідуальні цілі та відслідковувати прогрес, що є важливим для підтримки мотивації та досягнення результатів.

Інтерфейс MyFitnessPal наведено на рисунку 1.2.

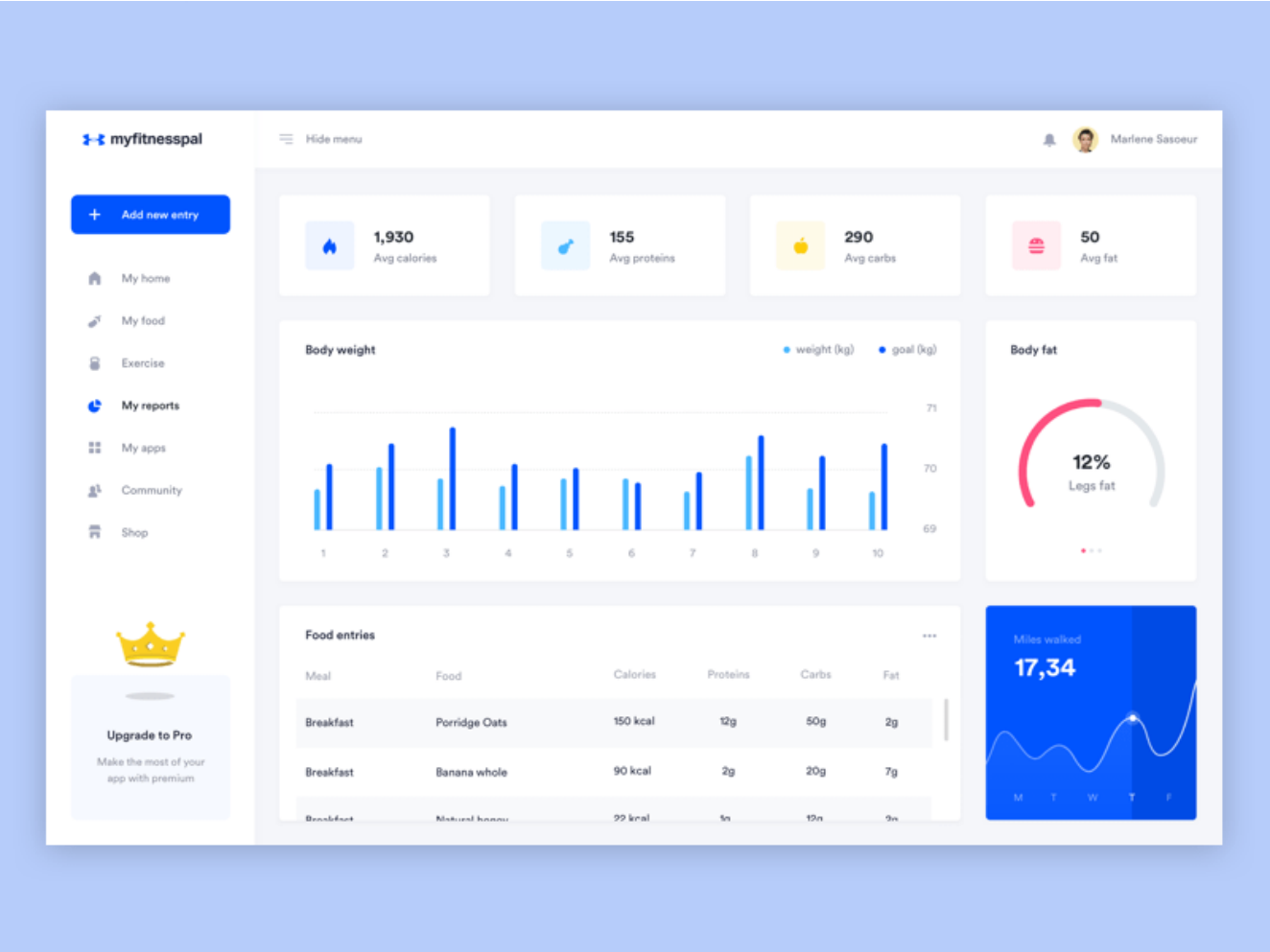


Рисунок 1.2 – Інтерфейс MyFitnessPal

Yazio [12] – це сучасний застосунок для планування раціону харчування та відстеження калорій, орієнтований на тих, хто прагне оптимізувати свій раціон та підтримувати здоровий спосіб життя. Додаток пропонує індивідуальні плани харчування, які враховують особисті цілі, вподобання та фізичну активність користувача. Yazio підходить як для тих, хто хоче схуднути, так і для тих, хто просто

прагне вести збалансоване харчування.

Основні можливості Yazio включають детальний аналіз споживаних калорій, макро- та мікронутрієнтів, а також можливість реєстрації прийомів їжі за допомогою простого та зручного інтерфейсу. Додаток надає доступ до бази даних з великою кількістю продуктів, рецептів і рекомендацій щодо здорового харчування, що дозволяє користувачам легко планувати свої прийоми їжі. Крім того, Yazio дозволяє встановлювати цілі та відслідковувати прогрес, що сприяє досягненню бажаних результатів.

Інтерфейс Yazio наведено на рисунку 1.3.

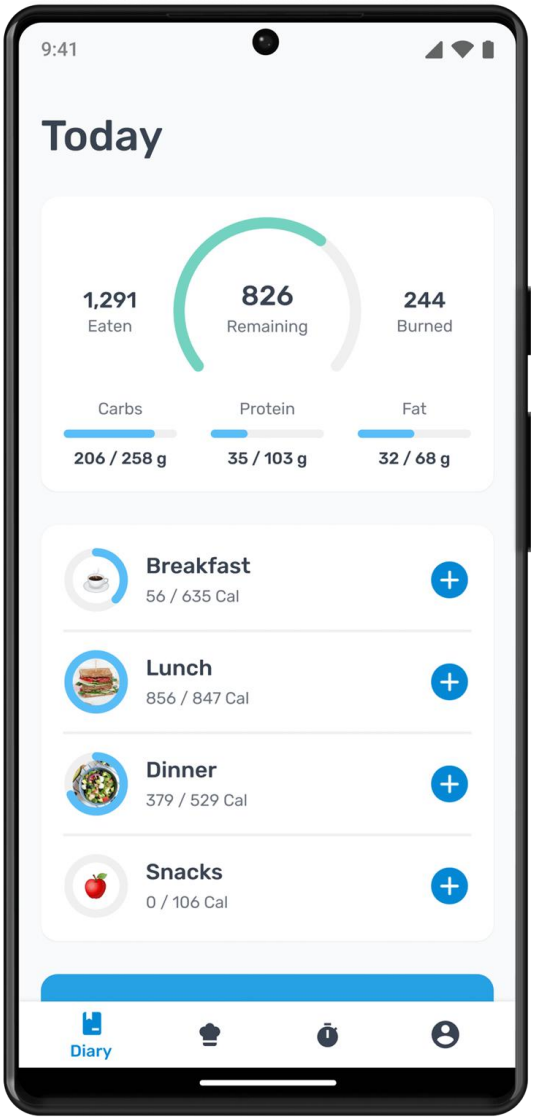


Рисунок 1.3 – Інтерфейс Yazio

Cronometer [13] – це застосунок для відстеження харчування, який дозволяє

користувачам отримувати глибокий аналіз споживання поживних речовин. Цей сервіс особливо корисний для тих, хто прагне детально контролювати не лише калорії, але й мікроелементи, вітаміни та мінерали, що споживаються щодня. Cronometer забезпечує точність даних, що робить його популярним серед спортсменів, дієтологів та осіб з особливими дієтичними потребами.

Інтерфейс Cronometer наведено на рисунку 1.4.

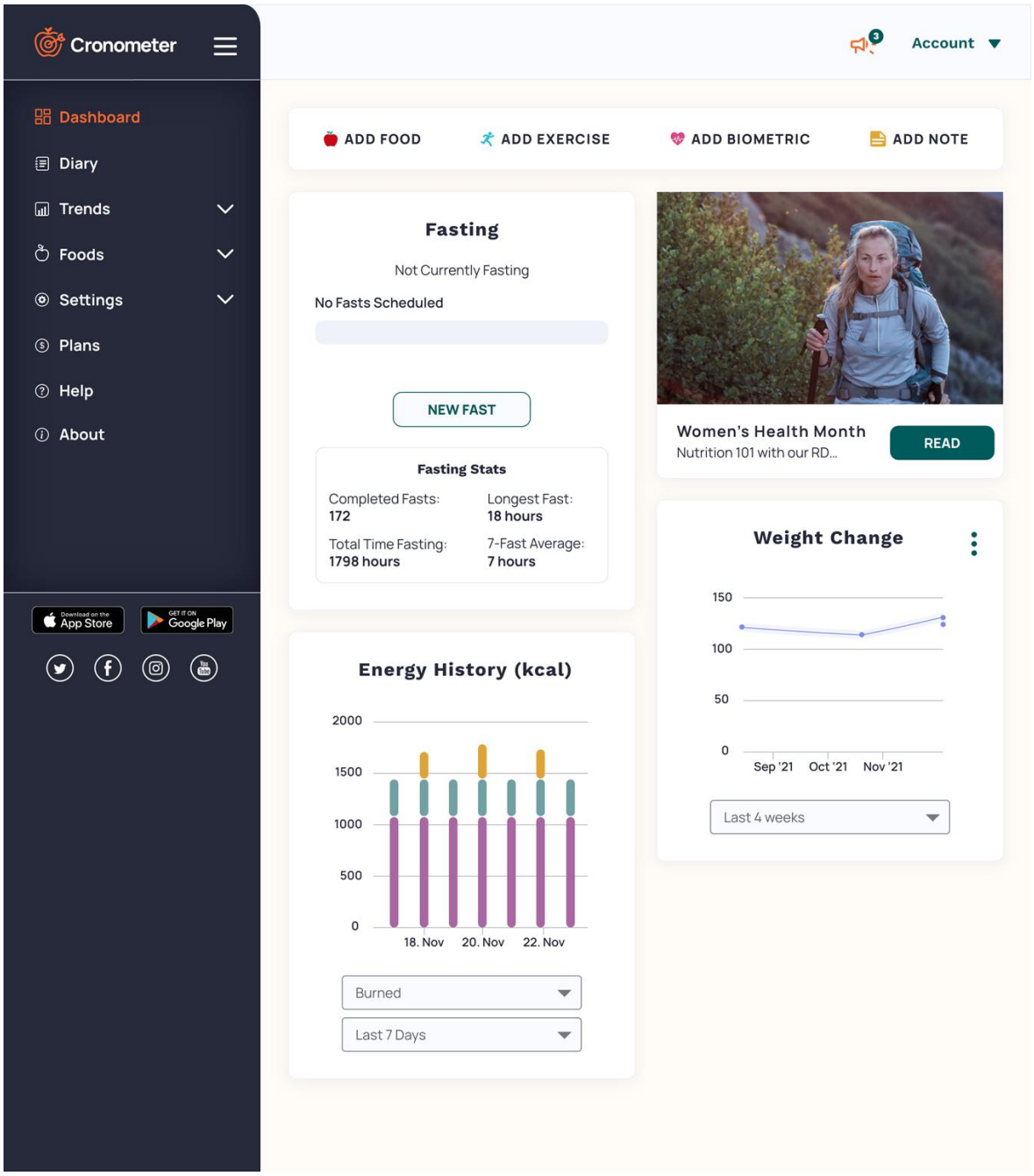


Рисунок 1.4 – Інтерфейс Cronometer

Основні функції Cronometer включають можливість введення та аналізу

даних про споживані продукти, що дозволяє відслідковувати баланс макро- та мікронутрієнтів. Додаток має зручний інтерфейс, який спрощує процес ведення харчового щоденника, а також забезпечує можливість інтеграції з іншими фітнес-додатками та носимими пристроями. Це дозволяє створити комплексну картину про загальний стан здоров’я користувача, включаючи показники фізичної активності, вагу та рівень енергії.

Таблиця 1.1 – Порівняльна таблиця характеристик застосунків для контролю здоров'я та харчування

Характеристика	Samsung Health	MyFitnessPal	Yazio	Cronometer	Власна розробка
Автоматичне розпізнавання їжі за фотографією	-	+	-	-	+
Динамічний перерахунок нутрієнтів за вагою порції	-	-	+	+	+
Персоналізовані рекомендації на основі даних харчування	+	+	+	+	+
Відстеження макронутрієнтів	+	+	+	+	+
Відстеження мікронутрієнтів	-	-	-	+	+
Відстеження фізичної активності	+	+	+	+	+
Деталізований аналіз складу страви	-	-	-	+	+
Сумарний бал	3	4	4	6	7

Таким чином, актуальною є власна розробка. Вона має вищий сумарний бал за Samsung Health на 57,1% (100% - 3/7*100% = 57,1%), за MyFitnessPal та Yazio на 42,9% (100% - 4/7*100% = 42,9%), за Cronometer на 14,2% (100% - 6/7*100% = 14,2%). Власна розробка має більш комплексний функціонал для контролю біологічних показників людини, об’єднує у собі переваги розглянутих аналогів.

1.3 Аналіз методів розв’язання задачі

Розглянемо відомі методи розв’язання задачі з розробки вебзастосунку для контролю біологічних показників людини.

Один з методів – використання монолітної архітектури, коли весь функціонал застосунку (збір, обробка даних, аналітика ШІ, інтерфейс користувача) реалізується у єдиному кодовому базі та працює на одному сервері чи кластері серверів [14]. Такий підхід простіший у розробці на початкових етапах і може бути зручним для невеликих проектів, де обсяг даних і навантаження незначні. Проте з ростом кількості користувачів та інтеграції нових джерел даних моноліт стає важчим для масштабування, його оновлення можуть порушувати роботу всього застосунку, а гнучкість у впровадженні інновацій – обмеженою.

Інший варіант – мікросервісна архітектура, при якій застосунок розбивається на незалежні модулі, кожен з яких відповідає за конкретну функцію: збір даних з носимих пристроїв, попередню обробку інформації, аналітику з використанням штучного інтелекту та інтерфейс користувача [15]. Такий підхід дозволяє легко масштабувати окремі сервіси, впроваджувати оновлення без порушення роботи інших компонентів і інтегрувати нові технології або джерела даних. Водночас, управління розподіленою системою потребує більш складної оркестрації, ефективного управління комунікаціями між сервісами та високого рівня тестування для забезпечення стабільності роботи.

Ще одним підходом є використання серверлес-технологій (serverless computing) або хмарних функцій для реалізації окремих частин застосунку. У цьому випадку окремі функції (наприклад, обробка потокових даних чи генерація рекомендацій штучного інтелекту) розгортаються як окремі серверлес-функції, що виконуються лише за потреби [16]. Такий підхід дозволяє знизити витрати на підтримку інфраструктури, автоматично масштабувати окремі компоненти та оптимізувати використання ресурсів. Проте існують певні виклики, пов'язані з холодним стартом функцій, складністю налагодження розподіленої логіки та обмеженнями у виконанні довготривалих завдань.

Серед розглянутих підходів оптимальним для розробки вебсервісу, що

контролює та оптимізує біологічні показники людини з використанням штучного інтелекту, є мікросервісна архітектура. Вона надає найбільшу гнучкість та масштабованість, дозволяє окремо оптимізувати компоненти системи, ефективно інтегрувати різноманітні джерела даних і швидко впроваджувати нові функції.

1.4 Постановка задач дослідження

Провівши аналіз методів розв'язання поставленої задачі, аналіз стану застосунків для контролю біологічних показників людини, аналізу аналогів було поставлено такі задачі дослідження:

– розробити архітектуру застосунку для контролю біологічних показників людини;

– розробити інтерфейс користувача застосунку;

– розробити алгоритми роботи застосунку;

– розробити функціонал застосунку;

– провести тестування розробленого застосунку.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було проведено аналіз сучасного стану застосунків для контролю біологічних показників людини, доведено актуальність програмної розробки в цьому напрямку, проведено порівняльний аналіз аналогів: Samsung Health, MyFitnessPal, Yazio, Cronometer, продемонстровано переваги власної розробки перед аналогами, проведено аналіз методів розв'язання задачі, поставлено задачі дослідження на розробку вебзастосунку для контролю біологічних показників людини з використанням штучного інтелекту.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1 Аналіз інформаційного забезпечення

Інформаційне забезпечення застосунку для контролю біологічних показників людини з використанням штучного інтелекту характеризується структурою даних, що забезпечує повний цикл моніторингу харчування від розпізнавання страв до формування довгострокової аналітики. Розглянемо детально типи даних, що використовуються в системі на різних етапах її функціонування.

Вхідні дані, що надаються користувачем, включають первинну фотографію страви у форматі цифрового зображення. Ці зображення характеризуються різною роздільною здатністю та якістю, що впливає на точність подальшого аналізу. Користувач також вводить інформацію про кількість спожитої їжі в грамах, що є критичним параметром для коректного розрахунку фактичного споживання нутрієнтів.

Окрім даних про окремі прийоми їжі, користувач надає системі персональну інформацію, включаючи антропометричні показники (вага, зріст), демографічні дані (вік, стать) та параметри активності. Додатково користувач встановлює цільові показники, такі як бажана вага або норми споживання певних нутрієнтів, що в подальшому використовуються для формування рекомендацій. При роботі застосунку відбувається обробка зображень з використанням методів комп'ютерного зору, формуються проміжні дані, що включають векторні представлення зображень, результати класифікації страв та ідентифіковані компоненти страви. Ці технічні дані не відображаються для користувача, але є критичними для функціонування системи.

В результаті аналізу зображення застосунок формує структуровану інформацію про страву, що включає назву розпізнаної страви та детальні кількісні дані про поживні речовини. Система визначає: калорійність, вміст білків, жирів, вуглеводів, клітковини, а також інформацію про вітаміни та мінерали. Ці дані представляються у стандартизованому форматі на 100 грамів продукту.

Після введення користувачем фактичної ваги порції система генерує масштабовані дані про нутрієнти, що відображають реальне споживання поживних речовин. Формуються як абсолютні значення, так і відсоткове співвідношення до рекомендованої добової норми споживання кожного нутрієнта. При збереженні запису про прийом їжі система генерує дані, що включають: часову мітку, категорію прийому їжі (сніданок, обід, вечеря, перекус) та зв'язок з профілем користувача.

Ці супутні дані забезпечують можливість формування хронологічної картини харчування та проведення часової аналітики.

На основі накопичених даних застосунок формує аналітичну інформацію різного рівня агрегації. Генеруються добові зведення про споживання нутрієнтів, тижневі та місячні тренди, а також порівняльна аналітика відносно встановлених цілей. Ця похідна інформація візуалізується у вигляді графіків та діаграм для зручного сприйняття користувачем.

В контексті фізичної активності система оперує даними про тренування, включаючи тип активності, тривалість, інтенсивність та розрахункові показники витрачених калорій.

Ці дані використовуються для коригування загального енергетичного балансу та формування комплексної картини здоров'я користувача. Інформаційне забезпечення застосунку також включає довідникові дані про стандартні норми споживання нутрієнтів, калорійність типових страв та коефіцієнти перерахунку для різних показників. Ці опорні дані використовуються як для внутрішніх розрахунків, так і для надання рекомендацій користувачеві.

Таким чином, інформаційне забезпечення застосунку представляє собою комплексну систему взаємопов'язаних даних, що охоплює весь цикл відстеження харчування – від розпізнавання окремих страв до формування цілісної картини харчових звичок та їх впливу на здоров'я користувача.

2.2 Розробка моделі системи

Модель системи розроблена у три рівні, які включають компоненти на стороні клієнта, модулі обробки на стороні сервера та механізми зберігання даних, доповнені інтеграцією зовнішньої служби та інфраструктурою безпеки.

Клієнтська частина охоплює п’ять основних компонентів, які спільно забезпечують роботу користувача. Компонент інтерфейсу користувача, створений за допомогою HTML5, CSS3 і TypeScript, служить рівнем презентації, відповідальним за відтворення всіх візуальних елементів, включаючи форми, інтерактивні елементи керування та відображення даних. Цей компонент підтримує постійний зв’язок із компонентом керування станом, який використовує унікальний JavaScript із модульною архітектурою для координації стану програми, обробки переходів між режимами та обробки взаємодії з користувачем.

Для можливостей обробки зображень компонент диспетчера зображень використовується технологія TypeScript File API і Canvas API для полегшення завантаження зображень, візуалізації попереднього перегляду та попередньої обробки перед передачею на сервер. Керування мережевим зв’язком здійснюється через компонент мережевого рівня, який реалізує Fetch API з асинхронними шаблонами TypeScript для встановлення надійних каналів зв’язку клієнт-сервер, передачі запитів HTTP та обробки вхідних відповідей.

Компонент візуалізації даних доповнює клієнтську архітектуру, використовуючи CSS і TypeScript DOM для створення та динамічного оновлення таблиць поживних речовин, діаграм і графічних зображень харчових даних, надаючи користувачам вичерпні результати аналізу в доступному форматі.

Серверна частина складається з п’яти компонентів, які виконують різні функції обробки. Компонент вебсервера, реалізований за допомогою Node.js і Express.js, служить точкою входу для всіх запитів HTTP, керує маршрутизацією, обробкою сеансів і координує взаємодію між клієнтськими запитами та ресурсами сервера.

Розроблена модель системи наведена на рисунку 2.1.

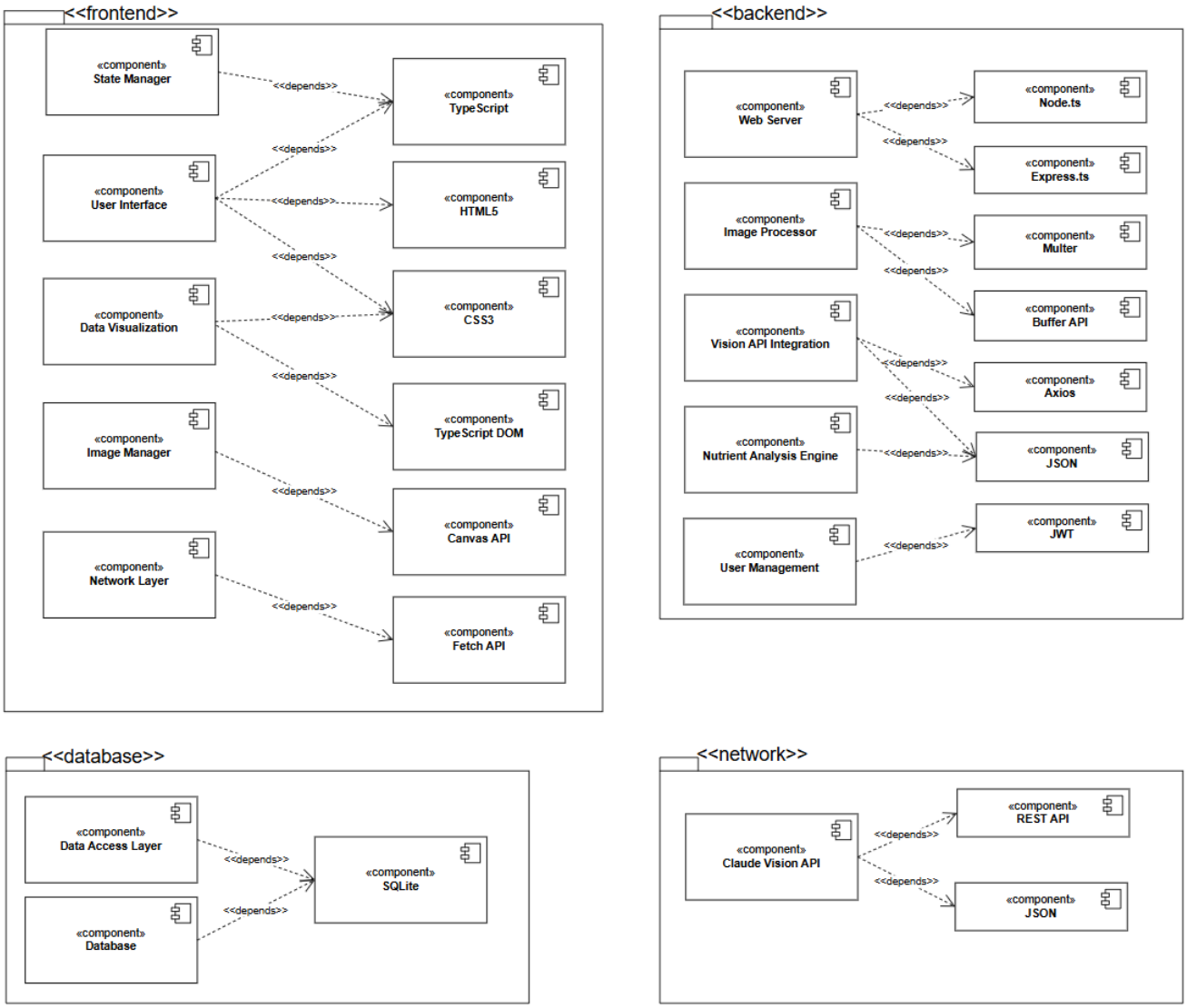


Рисунок 2.1 – Модель системи

Можливості обробки зображень надаються компонентом процесора зображень, який використовує Multer і Node.js Buffer API для керування прийомом, зберіганням, попередньою обробкою та перетворенням base64 для сумісності з API. Працюючи разом із цим компонентом, інтеграційний компонент Vision API використовує технології Axios і JSON для створення належним чином відформатованих запитів до Claude Vision API, перевірки відповідей і отримання відповідної інформації про харчування.

Компонент механізму аналізу поживних речовин обробляє отримані дані, виконуючи обчислення на основі ваги порції, стандартизуючи вихідні формати та готуючи дані для представлення та зберігання.

Рівень доступу до даних абстрагує операції з базою даних за допомогою SQLite з пакетами sqlite3 і sqlite Node.js, реалізуючи операції CRUD, керування транзакціями та надаючи єдиний інтерфейс для доступу до даних у програмі. Базовий компонент бази даних, створений на основі технології SQLite, забезпечує фізичне зберігання всіх системних даних, включаючи профілі користувачів, записи про харчування, інформацію про фізичні вправи та налаштування конфігурації.

Інтеграція зовнішніх служб полегшується через службу Claude Vision API, яка надає можливості розпізнавання зображень їжі через REST API з обміном даними JSON, що забезпечує основну функцію аналізу їжі програми.

Розроблена модель системи представляє структурований і модульний підхід до створення програми аналізу харчових продуктів із чіткими обов'язками компонентів, визначеними моделями взаємодії та відповідним вибором технологій для кожної функціональної області.

2.3 Розробка структури застосунку

Застосунок для контролю біологічних показників людини з використанням штучного інтелекту має трирівневу архітектуру, що включає клієнтську частину, серверну логіку та базу даних. Така архітектура забезпечує чіткий розподіл відповідальності між компонентами та спрощує подальше масштабування функціональності.

Клієнтська частина застосунку реалізована у формі інтерактивного вебінтерфейсу з використанням сучасних технологій розробки та структурована навколо основних функціональних блоків, що забезпечують взаємодію з користувачем.

JavaScript-модулі клієнтської частини забезпечують функціональну логіку, необхідну для коректної роботи інтерфейсу. Обробник завантаження зображень контролює процеси конвертації та передачі файлів на сервер. Модуль комунікації з сервером реалізує асинхронну взаємодію з API за допомогою технології AJAX.

Структура застосунку у вигляді дерева наведена на рисунку 2.2.

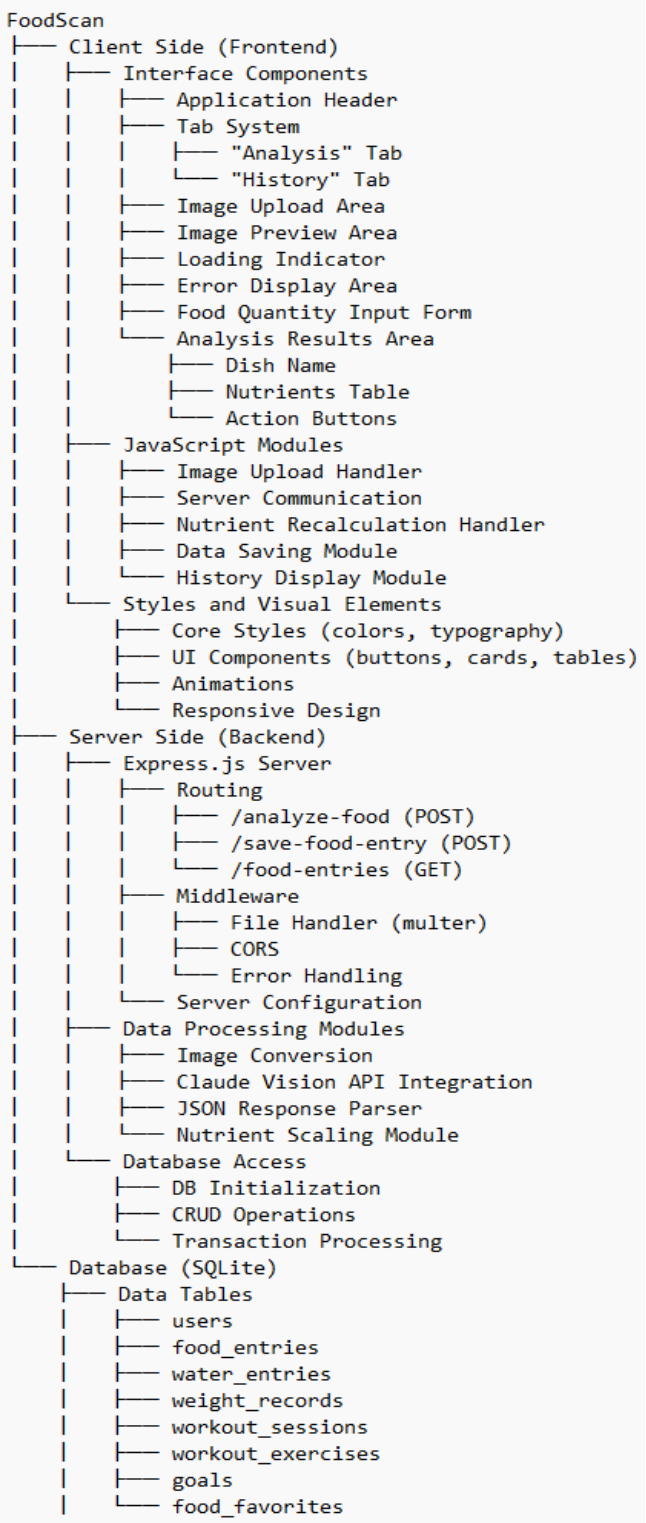


Рисунок 2.2 – Структура застосунку у вигляді дерева

Серверна частина застосунку побудована на базі платформи Node.js з використанням фреймворку Express.js. Система маршрутизації визначає три ключові кінцеві точки API: для аналізу зображень їжі, збереження записів про харчування та отримання історії. Middleware-компоненти забезпечують проміжну

обробку запитів, включаючи роботу з файлами через бібліотеку multer та підтримку крос-доменних запитів за допомогою CORS.

Модулі обробки даних на серверній стороні виконують спеціалізовані функції, що важливі для роботи системи. Модуль конвертації зображень трансформує завантажені файли у формат base64, необхідний для взаємодії з Claude Vision API. Модуль взаємодії з цим API реалізує формування коректних запитів та обробку отриманих відповідей. Парсер JSON-відповідей забезпечує структуризацію даних про нутрієнти для подальшого використання. Модуль масштабування реалізує алгоритми перерахунку нутрієнтів відповідно до введеної користувачем ваги порції.

Доступ до бази даних організовано через абстрактний шар, що забезпечує ініціалізацію з’єднання, виконання базових CRUD-операцій та коректну обробку транзакцій. Такий підхід ізолює логіку роботи з даними від бізнес-логіки застосунку, що спрощує потенційну міграцію на інші системи управління базами даних у майбутньому.

Структура застосунку спроектована з урахуванням принципів модульності та розширюваності, що дозволяє додавати нові функціональні можливості без суттєвої модифікації існуючих компонентів. Така архітектура забезпечує не лише поточні потреби системи, але й створює основу для подальшого розвитку застосунку в напрямку комплексного моніторингу здоров'я та персоналізованих рекомендацій щодо харчування.

2.4 Розробка методу автоматичного аналізу харчування на основі комп’ютерного зору

Метод комплексного аналізу та персоналізації нутрієнтного складу їжі на основі комп'ютерного зору поєднує технології комп'ютерного зору для ідентифікації страв з алгоритмами динамічного перерахунку поживних речовин відповідно до фактичної ваги порції. Метод забезпечує аналіз від розпізнавання зображення страви до формування точних даних про споживання нутрієнтів з урахуванням індивідуальних особливостей харчування користувача. Таке

поєднання дозволяє позбавляє необхідному ручного пошуку інформації та спрощує процес відстеження харчування.

Блок-схема алгоритму підготовки зображення до аналізу наведена на рисунку 2.3.

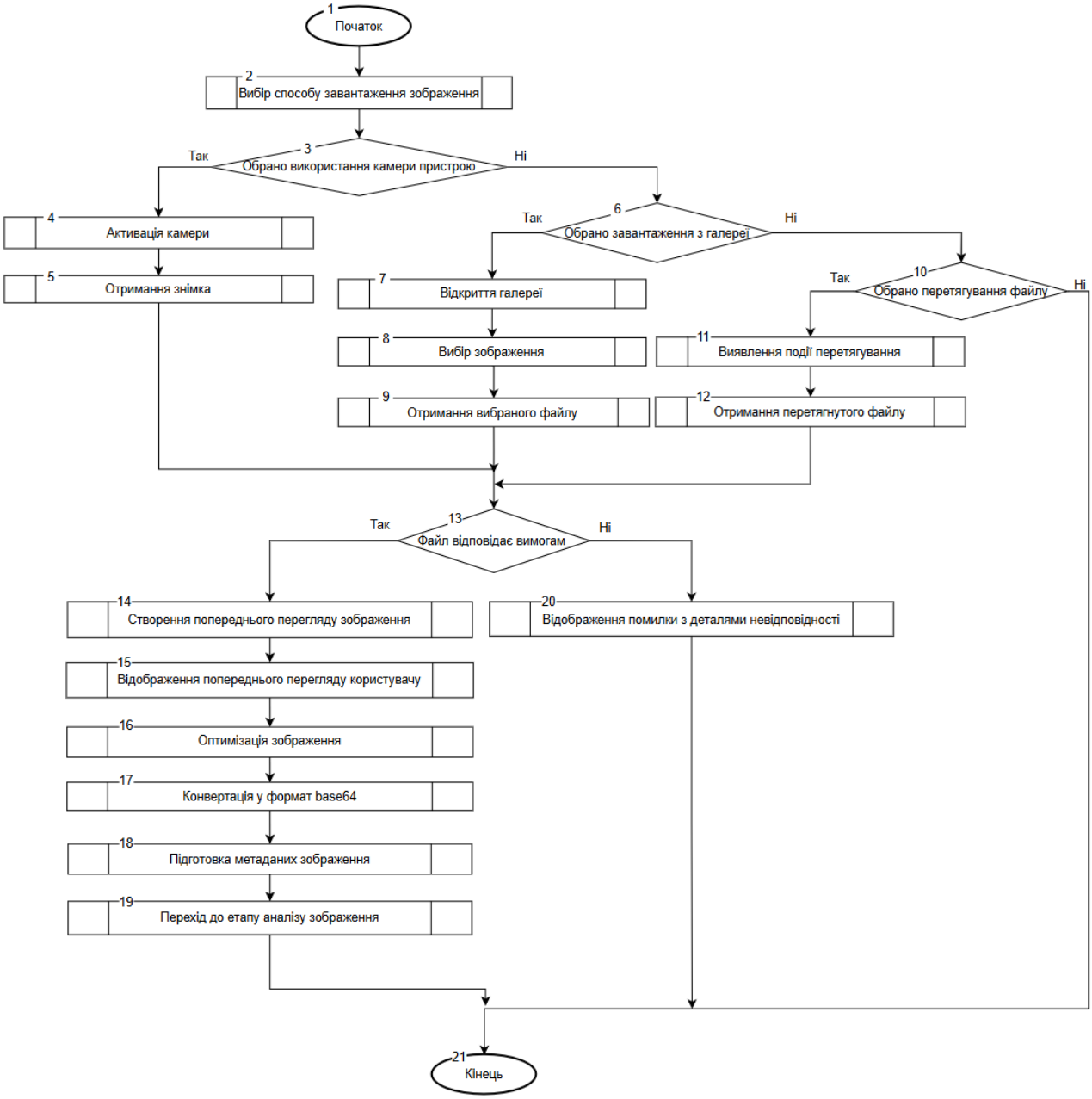


Рисунок 2.3 – Блок-схема алгоритму підготовки зображення до аналізу

Алгоритм методу:

1 . Користувач робить фотографію страви або завантажує існуюче зображення через інтерфейс застосунку.

2. Система конвертує зображення у формат, придатний для обробки (base64), та передає його до модуля комп'ютерного зору.

3. Модуль комп'ютерного зору, інтегрований з великою мовною моделлю (Claude Vision API), аналізує візуальні характеристики страви, включаючи колір, текстуру, форму компонентів та їх розташування. Блок-схема алгоритму цього процесу наведена на рисунку 2.4.

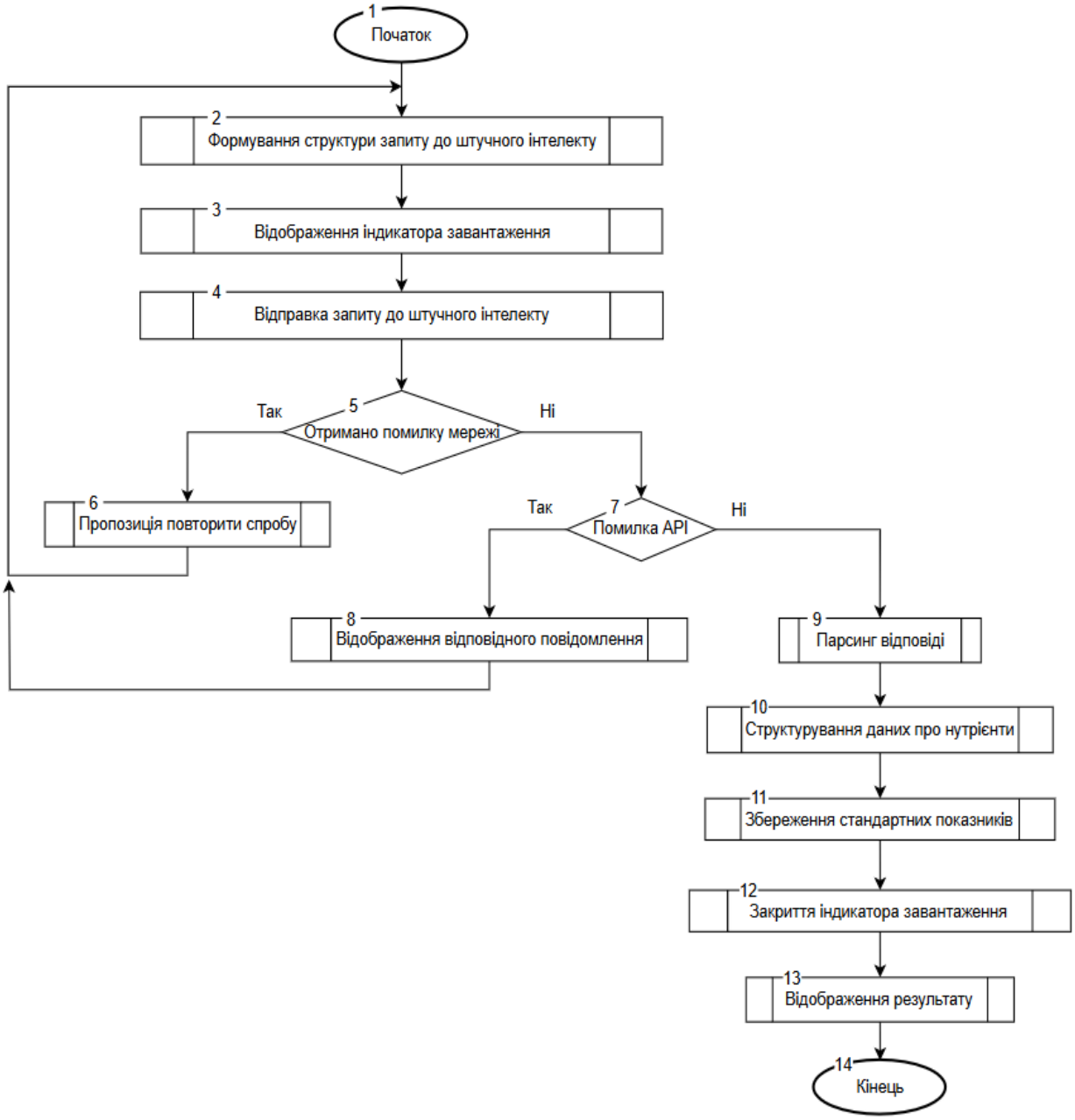


Рисунок 2.4 – Блок-схема алгоритму аналізу зображення

4. Система ідентифікує тип страви та визначає її стандартний нутрієнтний

склад на 100 грамів, включаючи калорійність, макронутрієнти (білки, жири, вуглеводи), мікронутрієнти (вітаміни, мінерали) та інші показники (клітковина, цукри, насичені жири).

5. Отримана інформація структурується у стандартизованому форматі JSON для подальшої обробки та зберігання.

6. Система запитує користувача про фактичну вагу спожитої порції або пропонує оцінити її на основі візуальних характеристик зображення та типових розмірів порцій для даної страви.

7. Алгоритм динамічного перерахунку застосовує математичну модель пропорційного масштабування до кожного нутрієнта за формулою:

$$нутрієнт\_факт = нутрієнт\_стандарт \times вага\_фактична / 100.$$

8. Система зберігає як початкові дані про нутрієнти (на 100 г), так і розраховані значення для фактичної порції, забезпечуючи можливість подальшого перерахунку без втрати точності.

9. На основі актуальних даних про споживання формується візуальне відображення нутрієнтного складу з урахуванням рекомендованих добових норм споживання.

10. Інформація про розпізнану страву та її нутрієнтний склад зберігається в базі даних з прив'язкою до часової мітки, що дозволяє формувати хронологічні звіти про харчування.

11. При кожній зміні введеної користувачем ваги порції система оперативно перераховує значення всіх нутрієнтів, зберігаючи високу точність даних.

12. На основі накопичених даних система може формувати аналітичні звіти про відповідність фактичного споживання нутрієнтів індивідуальним потребам користувача.

Метод включає модуль обробки зображень на основі сучасних бібліотек комп'ютерного зору, інтерфейс взаємодії з Claude Vision API для аналізу вмісту зображень і розпізнавання страв, та математичний модуль для динамічного перерахунку нутрієнтного складу з використанням оптимізованих алгоритмів.

Блок-схема алгоритму перерахунку нутрієнтів та подальшого збереження

даних наведена на рисунку 2.5.

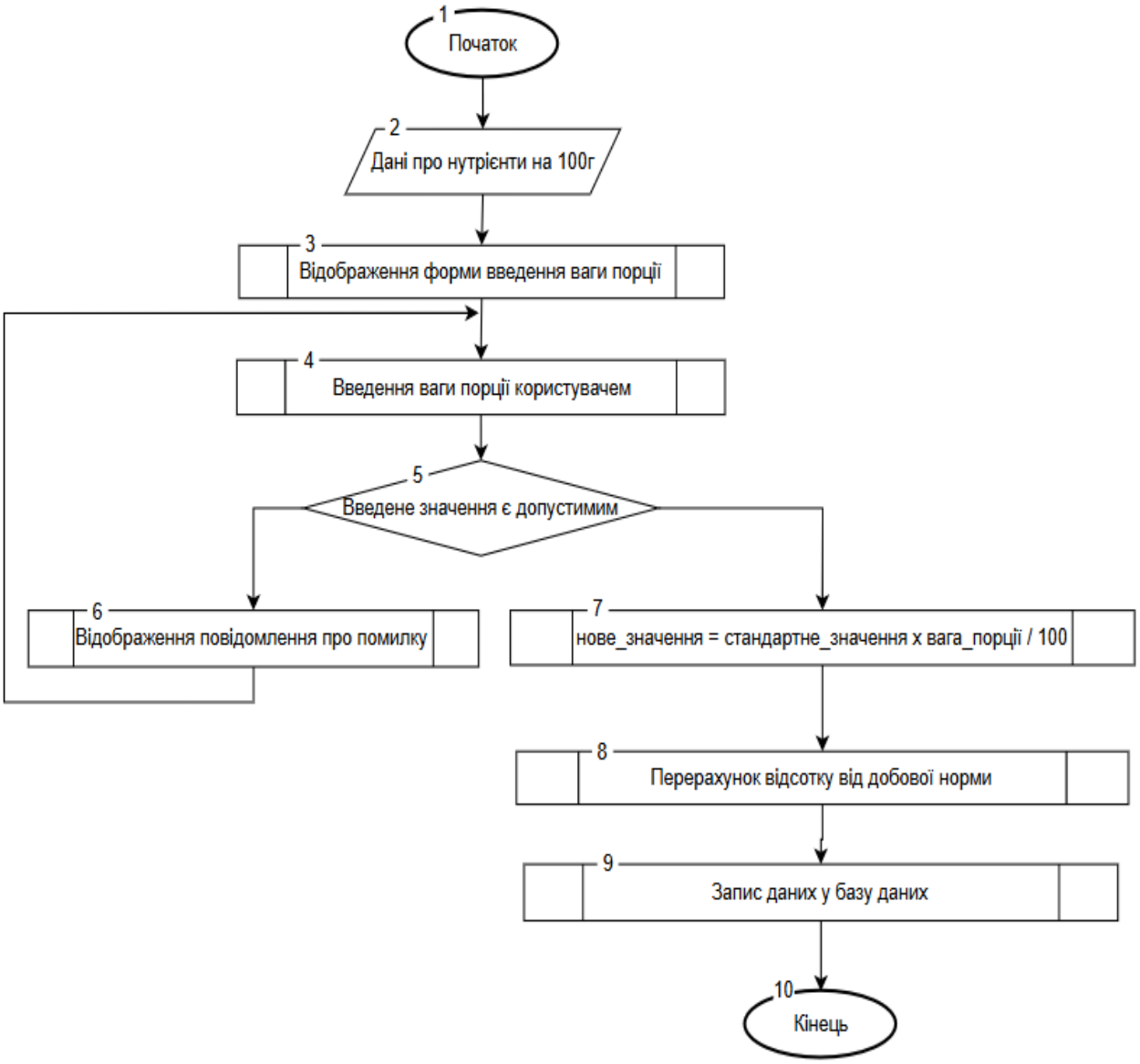


Рисунок 2.5 – Блок-схема алгоритму перерахунку нутрієнтів та збереження даних в базу даних

Цей метод має такі переваги:

1.Автоматизація процесу визначення складу їжі зменшує когнітивне навантаження на користувача та підвищує ймовірність регулярного відстеження харчування.

2.Точна персоналізація даних відповідно до фактичного розміру порції забезпечує достовірну оцінку споживання нутрієнтів.

3.Метод забезпечує високу точність даних навіть при варіативності розмірів порцій, що відповідає реальним практикам харчування.

2.5 Розробка методу персоналізованих адаптивних рекомендацій

Метод забезпечує формування індивідуалізованих рекомендацій через аналіз історичних даних про споживання їжі, води та тренувань, отриманих за допомогою комп'ютерного зору, та їх зіставлення з нутрієнтними потребами користувача, його цілями та медичними обмеженнями.

Алгоритм методу:

1. Система накопичує базу даних харчової поведінки користувача, створюючи детальний профіль користувача.

2. На основі антропометричних даних (вік, стать, вага, зріст) та рівня активності розраховуються індивідуальні потреби в калоріях, ключових нутрієнтах, та фізичних вправах.

3. Користувач визначає свої цілі (зниження ваги, набір м'язової маси, підтримка здоров'я) та вказує медичні обмеження (алергії, хронічні захворювання).

4. Алгоритм аналізує розбіжності між фактичним споживанням нутрієнтів та індивідуальними потребами, ідентифікуючи дефіцити та надлишки конкретних поживних речовин.

5. На основі даних профілю система ідентифікує харчові переваги користувача, визначаючи страви, які споживаються регулярно.

6. Формується набір рекомендованих продуктів та страв, що компенсують виявлені дефіцити нутрієнтів, відповідають цілям та обмеженням користувача, враховуючи його харчові вподобання.

7. Система використовує механізми прогресивного навчання для адаптації рекомендацій на основі зворотного зв'язку від користувача та даних про фактичне споживання рекомендованих страв.

Блок-схема процесу підготовки даних для формування рекомендацій наведена на рисунку 2.6.

Рисунок 2.6 – Блок-схема алгоритму процесу підготовки даних для формування рекомендацій

8. Через інтерфейс застосунку користувач отримує персоналізовані рекомендації щодо оптимальних страв, альтернативних продуктів та пропорцій прийомів їжі і фізичних вправ.

9. Періодичне порівняння цільових показників з фактичними результатами дозволяє системі оптимізувати стратегію рекомендацій для досягнення найкращого

результату.

Цей метод відрізняється від існуючих підходів до формування дієтичних рекомендацій інтеграцією об'єктивних даних про фактичне харчування, з адаптивними алгоритмами прогнозування, що забезпечує формування не лише теоретично оптимальних, але й практично реалізованих користувачем рекомендацій.

2.6 Висновки

У другому розділі було проведено аналіз інформаційного забезпечення вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту, визначено дані, які збираються, обробляються, генеруються при роботі вебсервісу. Розроблено модель системи у три шари – клієнтська частина, серверна та мережева. Описано компоненти, що використовуються у роботі вебсервісу. Розроблено архітектуру вебсервісу, побудовано деревовидну структуру, що містить усі класи, методи, таблиці для роботи вебсервісу. Розроблено методи автоматичного аналізу харчування на основі комп’ютерного зору та персоналізованих адаптивних рекомендацій.

3 РОЗРОБКА СИСТЕМИ ДЛЯ МОНІТОРИНГУ БІОЛОГІЧНИХ ПОКАЗНИКІВ ЛЮДИНИ І ПІДТРИМКИ СПОСОБУ ЖИТТЯ

3.1 Варіантний аналіз та вибір мови програмування розробки

Розглянемо такі мови програмування, як TypeScript, Python, Javascript.

TypeScript [17] – це статично типізована мова програмування з відкритим кодом, розроблена Microsoft з метою розширення можливостей JavaScript та боротьби з обмеженнями JavaScript у великих проектах, при цьому має сумісність з усім існуючим кодом JavaScript. Основна перевага мови полягає в статичній типізації, яка дозволяє виявляти поширені помилки на етапі компіляції, а не під час виконання програми.

TypeScript надає розробникам інструменти для визначення типів змінних, параметрів функцій, повертаємих значень та структур даних через інтерфейси і типи, що суттєво підвищує безпеку та надійність коду, особливо у великих проектах. TypeScript підтримує сучасні можливості ECMAScript, включаючи асинхронне програмування через async/await, що є важливо для розробки застосунків з інтенсивною обробкою даних та взаємодією з API. TypeScript компілюється у чистий JavaScript, що забезпечує виконання коду в будь-якому середовищі, де працює JavaScript – від браузерів до серверів на базі Node.js.

Python [18] – це високорівнева інтерпретована мова програмування з динамічною типізацією. Дизайн Python означає хорошу читабельність коду завдяки використанню значних відступів для структурування програми. Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтоване, імперативне, функціональне та процедурне програмування. Сильними сторонами Python є сфери наукових обчислень, аналізу даних та машинного навчання, де мова має розвинену екосистему бібліотек, таких як NumPy, Pandas, SciPy та TensorFlow. Для веброзробки Python підтримується фреймворками Django та Flask, що дозволяють створювати масштабовані вебзастосунки. Попри це, Python має обмежені можливості для клієнтської веброзробки, оскільки не виконується безпосередньо у браузерах. Динамічна типізація Python спрощує початковий процес розробки, але

може призводити до складнощів при масштабуванні великих проектів, де помилки типів виявляються лише під час виконання програми.

JavaScript [19] – це динамічна, інтерпретована мова програмування, яка є стандартною для веброзробки на стороні клієнта. JavaScript є мультипарадигмовою мовою, що підтримує об'єктно-орієнтований, імперативний та функціональний стилі програмування. З впровадженням Node.js у 2009 році, мова розширила свої можливості на серверну сторону, дозволяючи використовувати один і той же код на стороні клієнта та сервера. JavaScript характеризується динамічною типізацією та прототипним наслідуванням, що надає гнучкість, але може призводити до помилок при некоректному використанні типів даних. Основними перевагами JavaScript є універсальність, широка підтримка браузерами та велика спільнота розробників, що створила багату екосистему бібліотек та фреймворків, таких як React, Angular та Vue.js для фронтенду, та Express.js для бекенду.

Для порівняння можливостей розглянутих мов програмування, було сформновано таблицю 3.1.

Таблиця 3.1 – Порівняльна таблиця характеристик мов програмування

Характеристика	TypeScript	Python	JavaScript
Підтримка ООП	+	+	+
Швидкість виконання клієнтської частини	+	-	+
Розвинена екосистема для веброзробки	+	+	+
Вбудована валідація типів даних	+	-	-
Інтеграція з Claude Vision API	+	+	+
Можливість повторного використання коду між frontend і backend	+	-	+
Підтримка асинхронного програмування	+	+	+
Строгий контроль помилок на етапі компіляції	+	-	-
Сумарний бал	8	4	6

Таким чином, TypeScript є оптимальним вибором для розробки вебзастосунку для моніторингу біологічних показників людини і підтримки здорового способу життя. Статична типізація TypeScript забезпечує значні переваги при розробці

системи з комплексною обробкою даних. Враховуючи, що застосунок працює з різноманітними типами даних (зображення, структуровані дані про нутрієнти, користувацькі налаштування), TypeScript допомагає запобігти типовим помилкам на етапі розробки, а не під час виконання.

На відміну від Python, TypeScript забезпечує нативну інтеграцію з вебтехнологіями та може виконуватися безпосередньо у браузері, що є важливим для інтерактивного застосунку з обробкою зображень та динамічним відображенням результатів.

На відміну від звичайного JavaScript, TypeScript додає шар надійності через статичну типізацію, що особливо важливо для застосунку, який працює з критичними даними про харчування, що можуть впливати на здоров'я користувачів.

Таким чином, TypeScript надає баланс між продуктивністю розробки, надійністю коду та інтеграцією з вебтехнологіями.

3.2 Розробка бази даних

Для ведення обліку зміни ваги, спожитої їжі, її нутрієнтів, спожитої води та тренувань, необхідна база даних, яка зберігатиме та оброблятиме ці дані, і надаватиме користувачеві за його потреби [20]. Тому було прийнято рішення розробити базу даних.

Таблиця users зберігає інформацію про користувачів системи. Вона є центральною у базі даних, оскільки більшість інших таблиць посилаються на неї через зовнішні ключі. Поля таблиці users наведені у таблиці 3.2.

Таблиця 3.2 – Поля таблиці users

Поле	Тип	Опис
id	INTEGER	Первинний ключ
username	TEXT	Унікальне ім'я користувача
email	TEXT	Електронна пошта (унікальна)

Продовження таблиці 3.2

Поле	Тип	Опис
password_hash	TEXT	Хеш пароля
first_name	TEXT	Ім'я користувача
last_name	TEXT	Прізвище користувача
birth_date	DATE	Дата народження
gender	TEXT	Стать
height	INTEGER	Зріст у см
created_at	DATETIME	Дата створення акаунту
last_login	DATETIME	Дата останнього входу

Таблиця food_entries зберігає записи про всю їжу, спожиту користувачами, включаючи детальну інформацію про харчову цінність. Поля таблиці food_entries наведені у таблиці 3.3.

Таблиця 3.3 – Поля таблиці food_entries

Поле	Тип	Опис
id	INTEGER	Первинний ключ
user_id	INTEGER	Зовнішній ключ на users.id
food_name	TEXT	Назва страви
amount_grams	INTEGER	Кількість спожитої їжі в грамах
meal_type	TEXT	Тип прийому їжі (сніданок, обід, вечеря, перекус)
calories	REAL	Калорійність
proteins	REAL	Вміст білків у грамах
fats	REAL	Вміст жирів у грамах
carbs	REAL	Вміст вуглеводів у грамах
fiber	REAL	Вміст клітковини у грамах
vitamins	TEXT	Інформація про вітаміни (JSON)
minerals	TEXT	Інформація про мінерали (JSON)

Таблиця water_entries містить записи про споживання води кожним користувачем. Поля таблиці water_entries наведені у таблиці 3.4.

Таблиця 3.4 – Поля таблиці water_entries

Поле	**Тип**	**Опис**
id	INTEGER	Первинний ключ
amount_ml	INTEGER	Кількість випитої води в мл
entry_datetime	DATETIME	Дата і час запису

Таблиця weight_records створена для ведення історії змін ваги та інших параметрів тіла користувача. Поля таблиці weight_records наведені у таблиці 3.5.

Таблиця 3.5 – Поля таблиці weight_records

Поле	**Тип**	**Опис**
id	INTEGER	Первинний ключ
user_id	INTEGER	Зовнішній ключ на users.id
weight_kg	REAL	Вага в кг
body_fat_percentage	REAL	Відсоток жиру в тілі (необов’язково)
muscle_mass_kg	REAL	Маса м'язів в кг (необов’язково)
measurement_date	DATETIME	Дата і час вимірювання
notes	TEXT	Додаткові нотатки

Таблиця workout_sessions зберігає загальну інформацію про тренування користувача та іншої активності. Поля таблиці workout_sessions наведені у таблиці 3.6.

Таблиця 3.6 – Поля таблиці workout_sessions

Поле	Тип	Опис
id	INTEGER	Первинний ключ
user_id	INTEGER	Зовнішній ключ на users.id
workout_type	TEXT	Тип тренування
duration_minutes	INTEGER	Тривалість в хвилинах
calories_burned	INTEGER	Спалені калорії
intensity	TEXT	Інтенсивність
start_time	DATETIME	Час початку тренування
end_time	DATETIME	Час закінчення тренування

Таблиця workout_exercises створена для детальної інформації про вправи, виконані під час тренування. Поля таблиці workout_exercises наведені у таблиці 3.7.

Таблиця 3.7 – Поля таблиці workout_exercises

Поле	Тип	Опис
id	INTEGER	Первинний ключ
workout_id	INTEGER	Зовнішній ключ на workout_sessions.id
exercise_name	TEXT	Назва вправи
sets	INTEGER	Кількість підходів
reps	INTEGER	Кількість повторень
weight_kg	REAL	Вага в кг
duration_seconds	INTEGER	Тривалість в секундах
notes	TEXT	Додаткові нотатки

Таблиця goals створена для відстеження прогресу в досягненні поставлених користувачами цілей. Поля таблиці goals наведені у таблиці 3.8.

Таблиця 3.8 – Поля таблиці goals

Поле	Тип	Опис
id	INTEGER	Первинний ключ
user_id	INTEGER	Зовнішній ключ на users.id
goal_type	TEXT	Тип цілі (вага, вода, тренування, харчування)
target_value	REAL	Цільове значення
current_value	REAL	Поточне значення
start_date	DATE	Дата початку
target_date	DATE	Цільова дата
is_completed	BOOLEAN	Статус виконання

Таблиця food_favorites призначена для спрощення ведення щоденника харчування шляхом швидкого додавання часто вживаних продуктів (таблиця 3.9).

Таблиця 3.9 – Поля таблиці goals

Поле	**Тип**	**Опис**
id	INTEGER	Первинний ключ
user_id	INTEGER	Зовнішній ключ на users.id
food_name	TEXT	Назва страви
amount_grams	INTEGER	Стандартна кількість в грамах
calories	REAL	Калорійність
proteins	REAL	Вміст білків
fats	REAL	Вміст жирів
carbs	REAL	Вміст вуглеводів
other_nutrients	TEXT	Інші поживні речовини (JSON)
created_at	DATETIME	Дата створення запису

Схема бази даних наведена на рисунку 3.1.

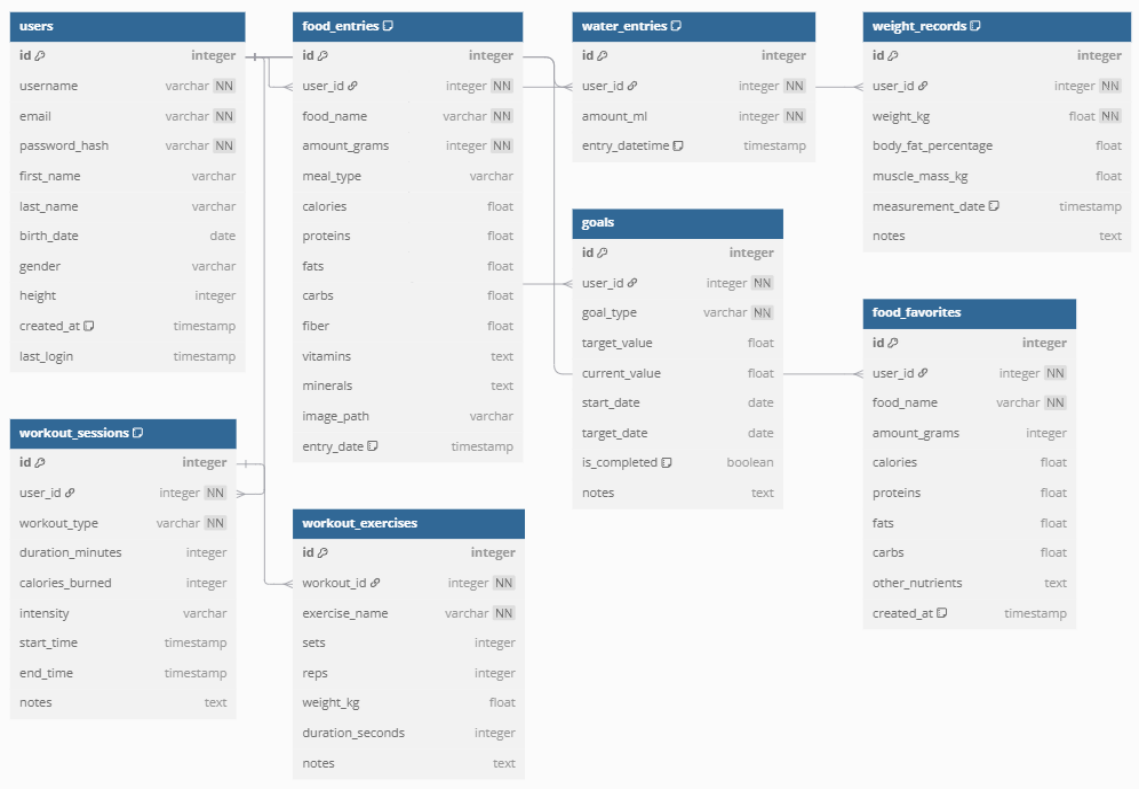


Рисунок 3.1 – Схема бази даних

Отже, розроблена база даних дозволяє зберігати дані про спожиту їжу, воду,

фізичні вправи, інформацію про користувача та його активність. Вона забезпечує роботу з даними для усього необхідного функціоналу застосунку для контролю біологічних показників людини з використанням штучного інтелекту.

3.3 Розробка інтерфейсу користувача

Інтерфейс користувача необхідний для зручної та зрозумілої взаємодії користувача із системою. Якісно спроєктований інтерфейс дозволяє користувачу легко опанувати функціонал та отримувати якісний користувацький досвід, що сприяє залученню більшої кількості користувачів [21].

Було спроєктовано основні вікна інтерфейсу користувача. Вікно налаштування профілю дозволяє вводити інформацію користувач про себе, таку як ім’я, зріст, цільову вагу, якої він хоче досягти чи підтримувати, а також споживання калорій. Ця інформація допоможе системі краще визначити рекомендації, які будуть корисні користувачу, а також допомагати у досягненні необхідної ваги та контролю харчування. Структурна схема інтерфейсу вікна налаштування профілю наведена на рисунку 3.2.

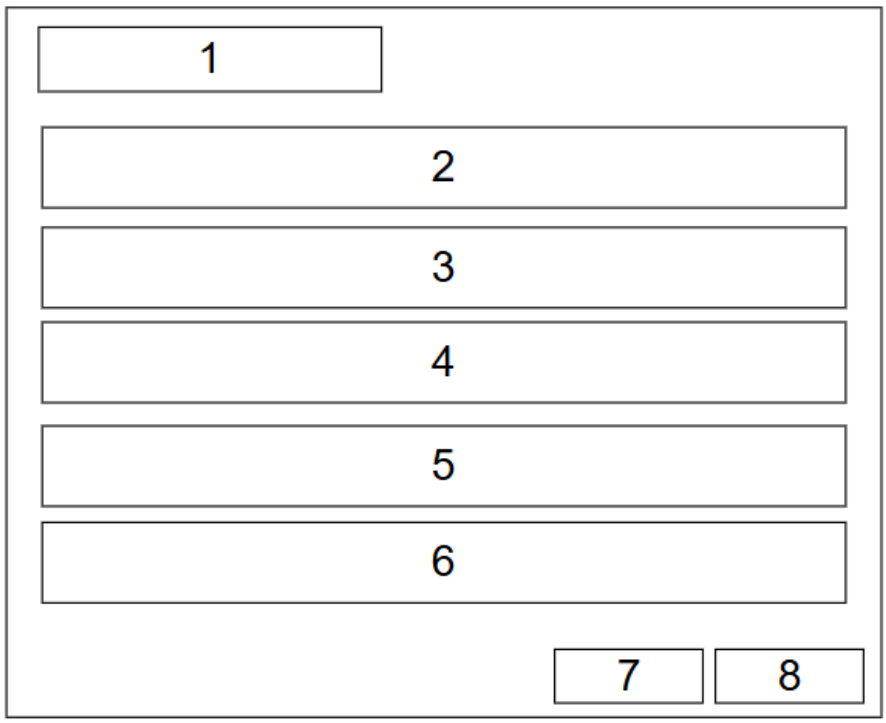


Рисунок 3.2 – Структурна схема інтерфейсу вікна налаштування профілю

Елементи вікна налаштування профілю:

1. Назва вікна.
2. Ім’я.
3. Зріст.
4. Цільова вага.
5. Цільові калорії на день.
6. Цільове споживання води.
7. Кнопка «Скасувати».
8. Кнопка «Зберегти».

Вікно додавання запису про вагу користувача дозволяє відстежувати зміни у вазі. Користувач може регулярно вносити відповідну інформацію у простому інтерфейсі, та отримувати графік з динамікою зміни ваги. Структурна схема інтерфейсу вікна налаштування профілю наведена на рисунку 3.3.

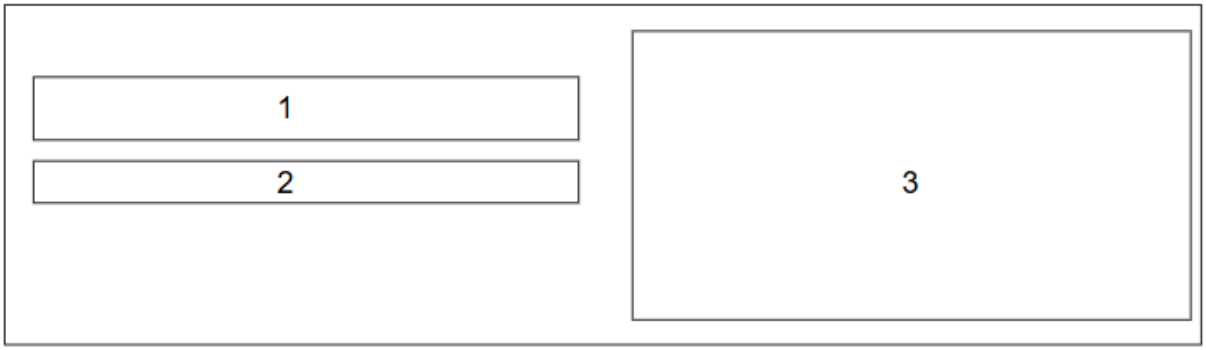


Рисунок 3.3 – Структурна схема вікна додавання запису про вагу

Елементи вікна додавання запису про вагу:

1. Поле для додавання ваги.
2. Остання вага.
3. Графік динаміки ваги.

Вікно відстеження спожитих калорій дозволяє відстежувати прогрес споживання калорій за день залежно від цільового показника, додавати кількість спожитих калорій вручну (якщо він має таку інформацію), або ж використати вбудований функціонал аналізу спожитих речовин з фото їжі. Після додавання

відображається список прийомів їжі за день. Структурна схема інтерфейсу вікна відстеження спожитих калорій наведена на рисунку 3.4.

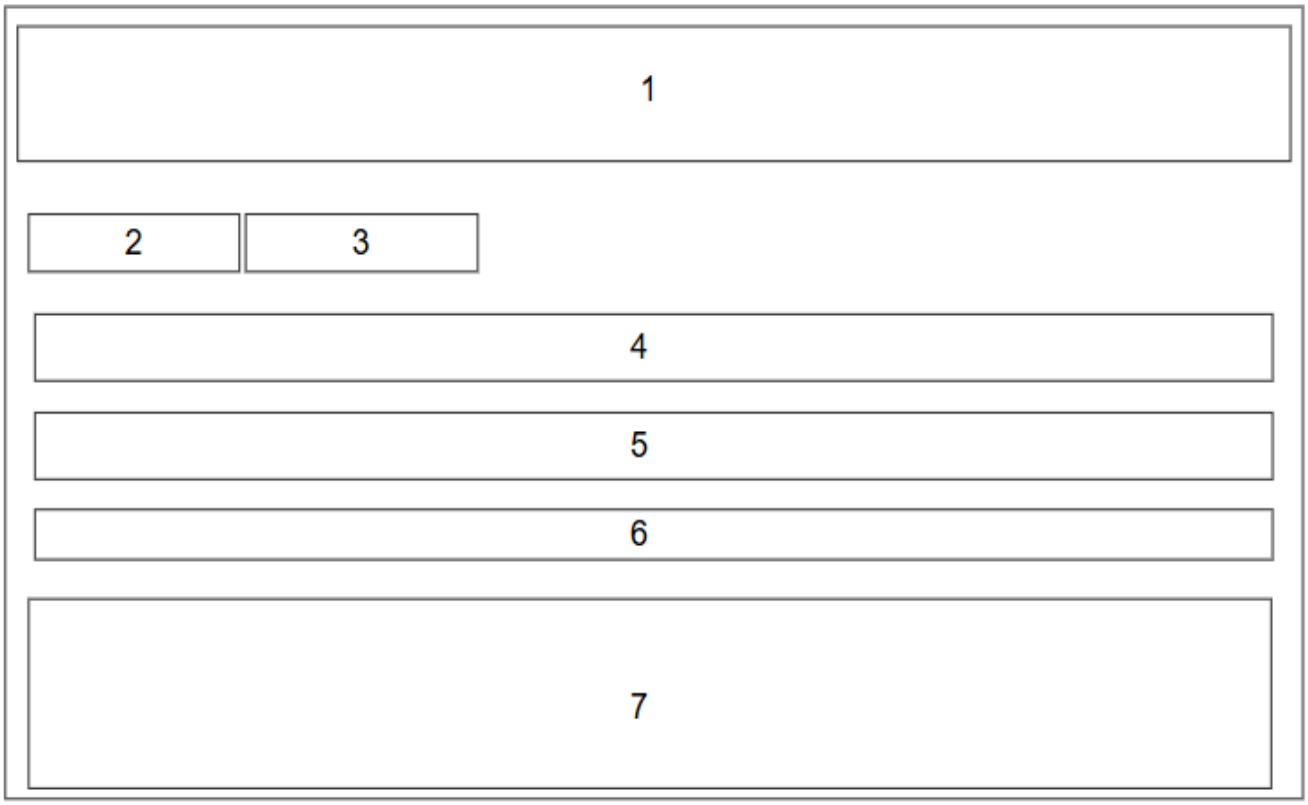


Рисунок 3.4 – Структурна схема інтерфейсу вікна відстеження спожитих калорій

Елементи вікна відстеження спожитих калорій:

1. Спожиті калорії за день.
2. Кнопка для режиму ручного додавання спожитих калорій.
3. Кнопка для режиму розпізнавання за фото.
4. Назва страви.
5. Калорії.
6. Кнопка «Додати».
7. Список прийомів їжі за день.

Вікно розпізнавання їжі за фото дозволяє завантажувати фото спожитої їжі, після чого за допомогою штучного інтелекту відбувається аналіз фото, який повертає результат у окремому вікні. Структурна схема інтерфейсу вікна розпізнавання їжі за фото наведена на рисунку 3.5.

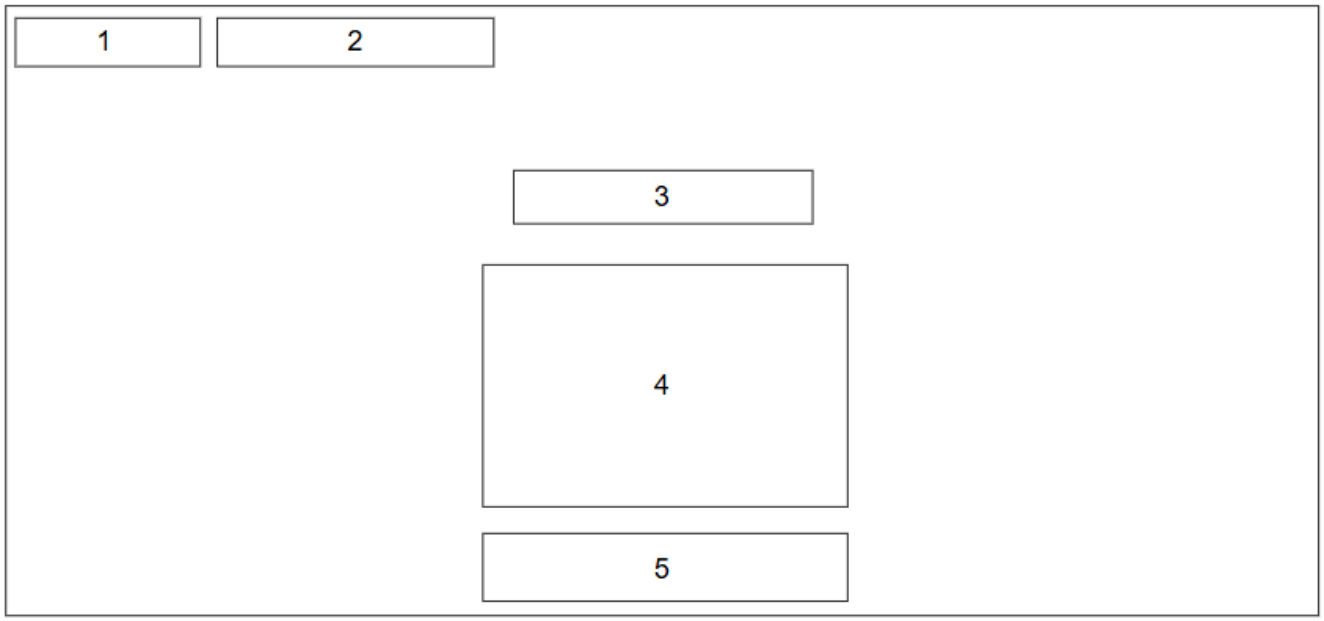


Рисунок 3.5 – Структурна схема інтерфейсу вікна розпізнавання їжі за фото

Елементи вікна розпізнавання їжі за фото:

1. Кнопка для режиму ручного додавання спожитих калорій.
2. Кнопка для режиму розпізнавання за фото.
3. Кнопка «Завантажити фото їжі».
4. Фото завантаженої їжі.
5. Надпис «Аналіз зображення».

Вікно інформації про спожиту їжу дозволяє користувачеві переглядати аналіз спожитої їжі за допомогою штучного інтелекту. Користувач отримує перелік калорій та речовин, що містяться у спожитій їжі, з можливістю вказати розмір порції, відповідно до чого відбудеться перерахунок спожитих нутрієнтів.

Елементи вікна інформації про спожиту їжу:

1. Назва їжі.
2. Поле для введення розміру порції.
3. Кнопка «Додати як прийом їжі».
4. Таблиця з інформацією про нутрієнти у їжі.

Структурна схема інтерфейсу вікна інформації про спожиту їжу наведена на рисунку 3.6.

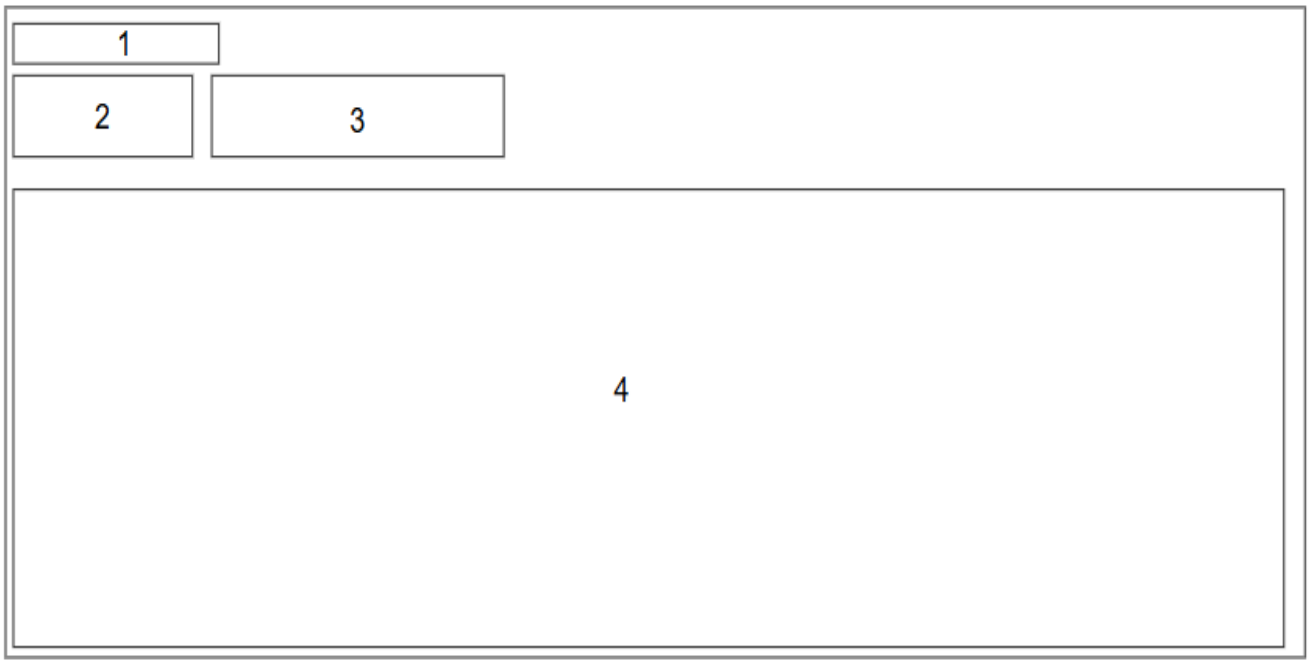


Рисунок 3.6 – Структурна схема інтерфейсу вікна інформації про спожиту їжу

Вікно записів про спожиту воду дозволяє окрім обліку їжі вести облік випитих напоїв, оскільки кількість спожитої води також впливає на вагу та стан організму. Це вікно містить поле для введення спожитої води, відстеження прогрес випитої води за день залежно від встановленого цільового показника, та переглядати історію споживання. Структурна схема інтерфейсу вікна записів про спожиту воду наведена на рисунку 3.7.

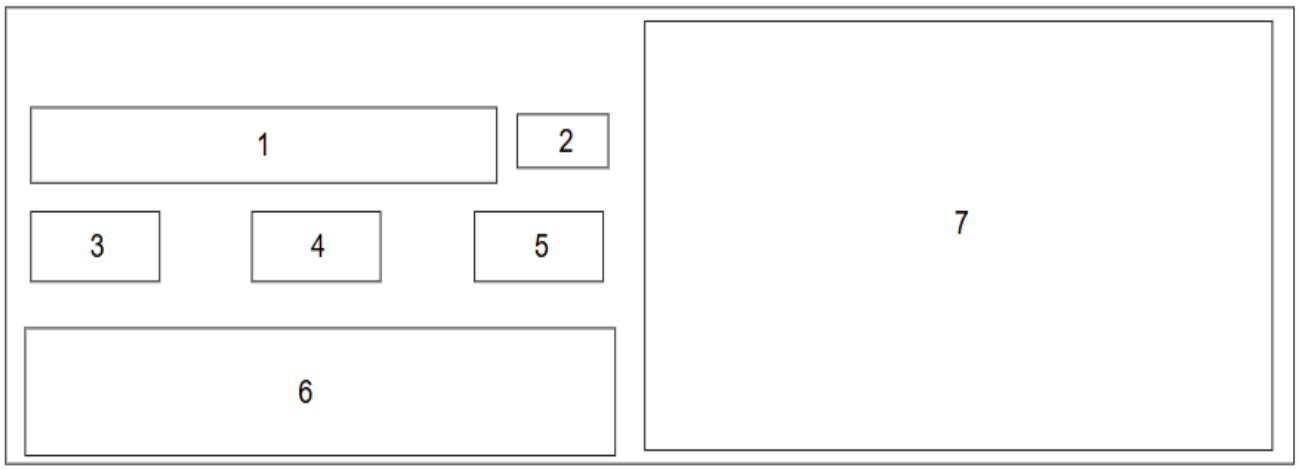


Рисунок 3.7 – Структурна схема інтерфейсу вікна записів про спожиту воду

Елементи вікна записів про спожиту воду:

1. Поле для введення кількості спожитої води.
2. Кнопка для додавання запису у базу даних.
3. Кнопка для додавання 200 мл води у базу даних.
4. Кнопка для додавання 300 мл води у базу даних.
5. Кнопка для додавання 500 мл води у базу даних.
6. Кількість спожитої води за сьогодні.
7. Записи спожитої води.

Вікно записів про тренування дозволяє вносити записи про проведені тренування, нотатки щодо тренувань, та переглядати список останніх тренувань. Крім цього, це вікно містить рекомендації щодо тренувань. Структурна схема інтерфейсу цього вікна наведена на рисунку 3.8.

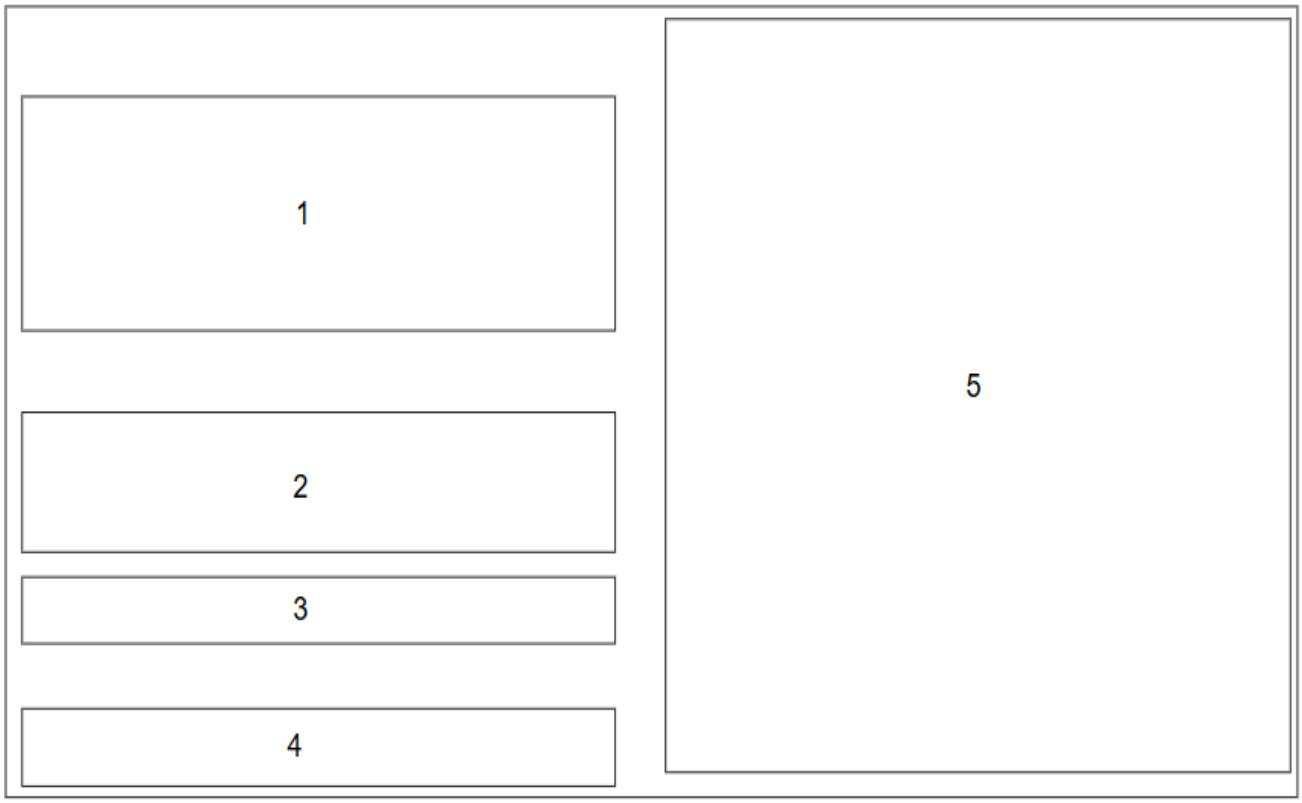


Рисунок 3.8 – Структурна схема інтерфейсу вікна записів про тренування

Елементи вікна записів про тренування:

1. Випадаючий список для типу тренування.

2. Нотатки.

3. Кнопка «Додати тренування».

4. Рекомендації щодо тренування.

5. Записи останніх тренувань.

Сторінка статистики аналізує дані про спожиті калорії, воду, виконані тренування та час сну за останній тиждень, та порівнює їх з попереднім тижнем, демонструючи різницю зміни у відсотках. Також міститься опис з детальним аналізом отриманих даних.

Структурна схема сторінки статистики наведена на рисунку 3.9.

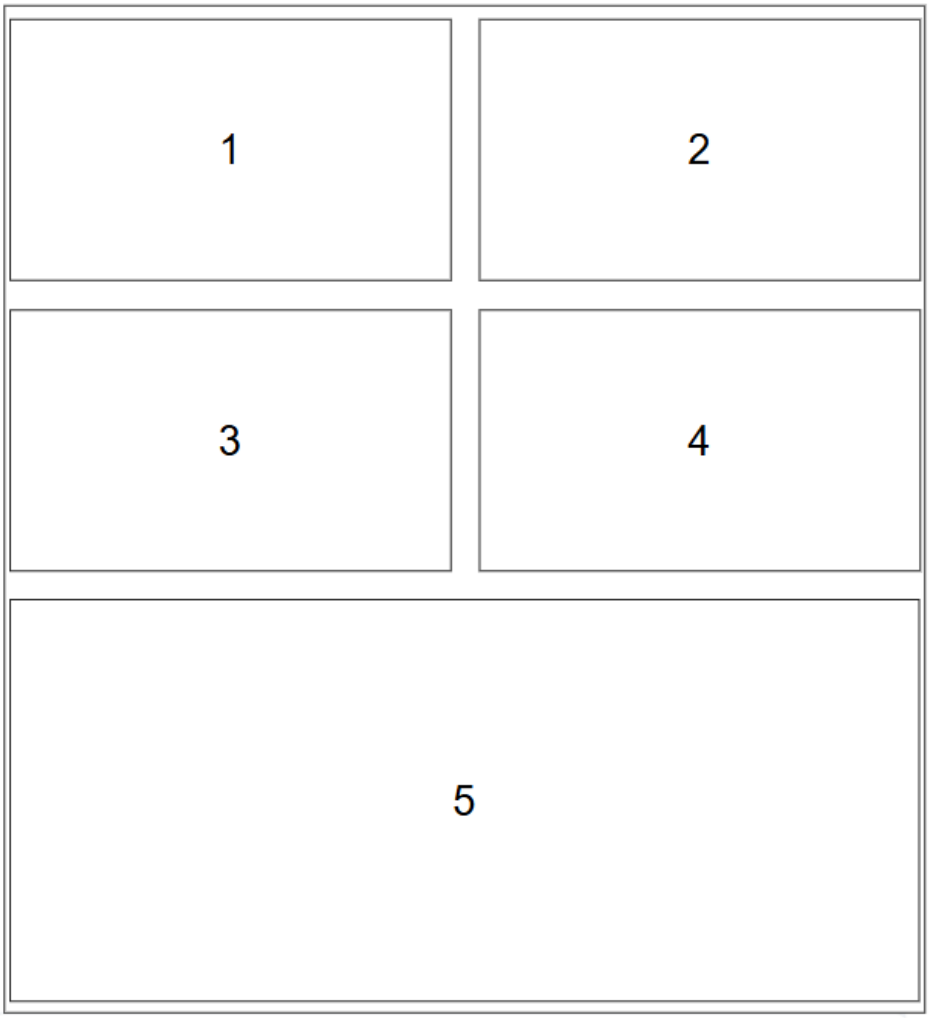


Рисунок 3.9 – Структурна схема сторінки статистики

Складові елементи сторінки статистики:

1. Середні калорії.

2. Спожита вода за день.

3. Кількість тренувань.

4. Середній сон.

5. Детальний аналіз.

Запис рекомендації дозволяє системі відображати рекомендації користувача, які базуються на основі наявних даних про його харчування, виконання вправ, кількість та якість сну, та загальний настрій.

Структурна схема інтерфейсу запису рекомендації наведена на рисунку 3.10.

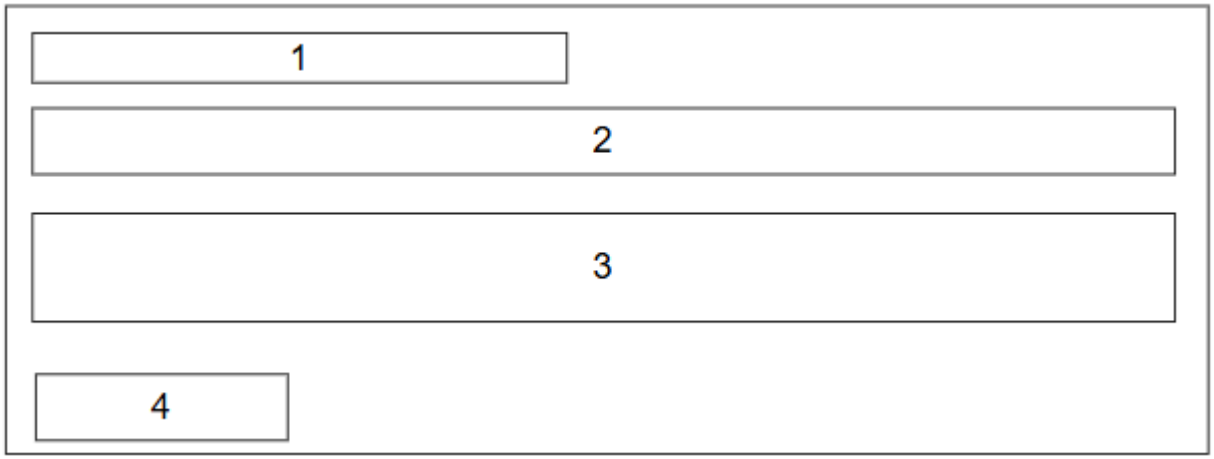


Рисунок 3.10 – Структурна схема інтерфейсу рекомендації

Складові елементи запису з рекомендаціями:

1. Назва рекомендації.

2. Детальний опис причини появи рекомендації.

3. Рекомендація.

4. Позначення виконання рекомендації.

Сторінка записів про сон дозволяє додавати інформацію про час початку та завершення сну, на основі чого обраховується тривалість сну. Також користувач може дати свою оцінку якості свого сну. Система при цьому відображає відповідні рекомендації на основі наявної інформації.

Структурна схема інтерфейсу запису про сон наведена на рисунку 3.11.

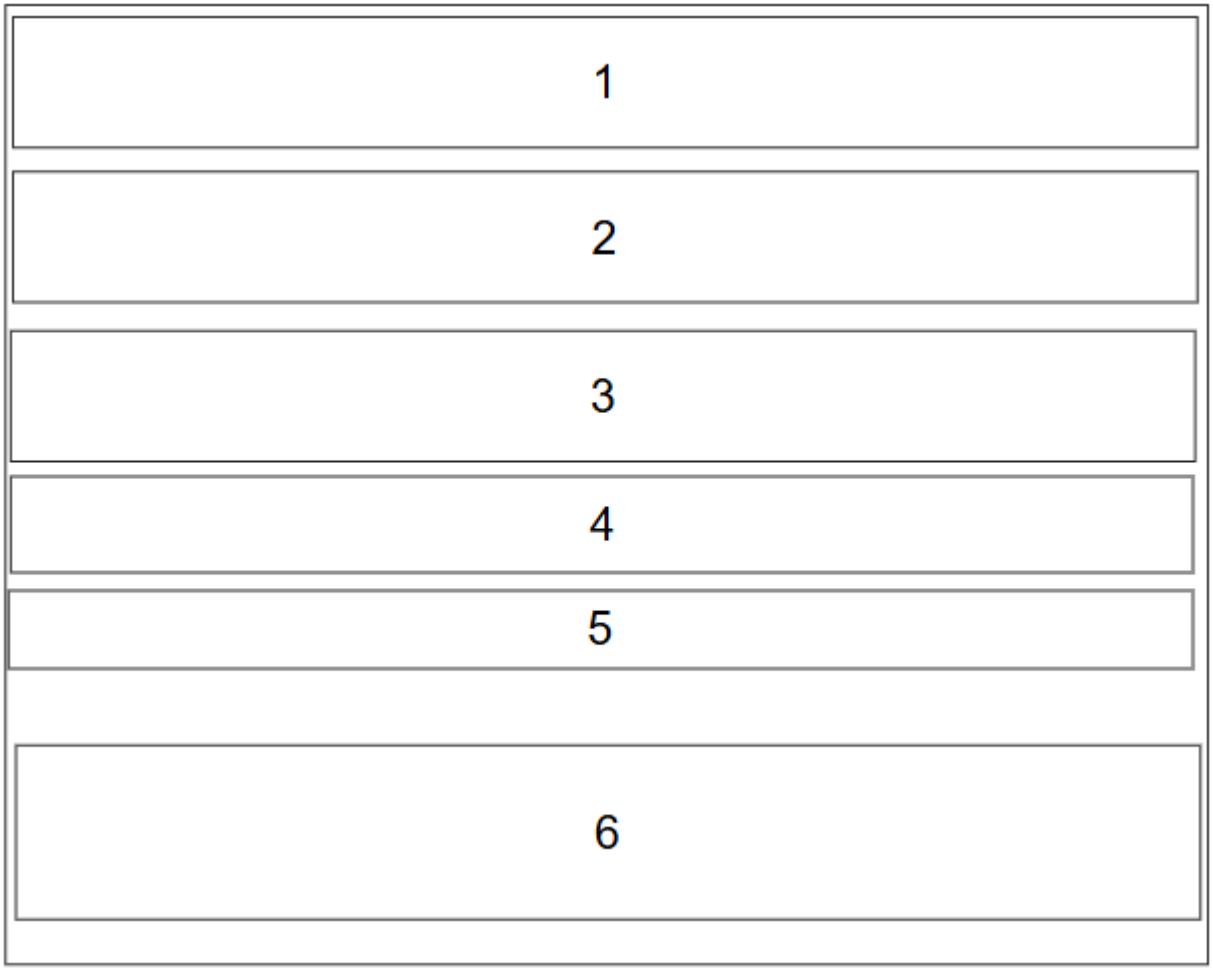


Рисунок 3.11 – Структурна схема сторінки запису про сон

Складові елементи сторінки запису про сон:

1. Час відходу до сну.
2. Час прокидання.
3. Тривалість сну.
4. Якість сну.
5. Кнопка «Зберегти дані про сон».
6. Персональні рекомендації.

Вікно детального аналізу самопочуття дозволяє користувачеві дати оцінку рівня енергії та стресу за шкалою від 1 до 0, вказати негативні фізичні симптоми, та записати додаткові нотатки щодо самопочуття.

Структурна схема інтерфейсу діалогового вікна детального аналізу самопочуття наведена на рисунку 3.12.

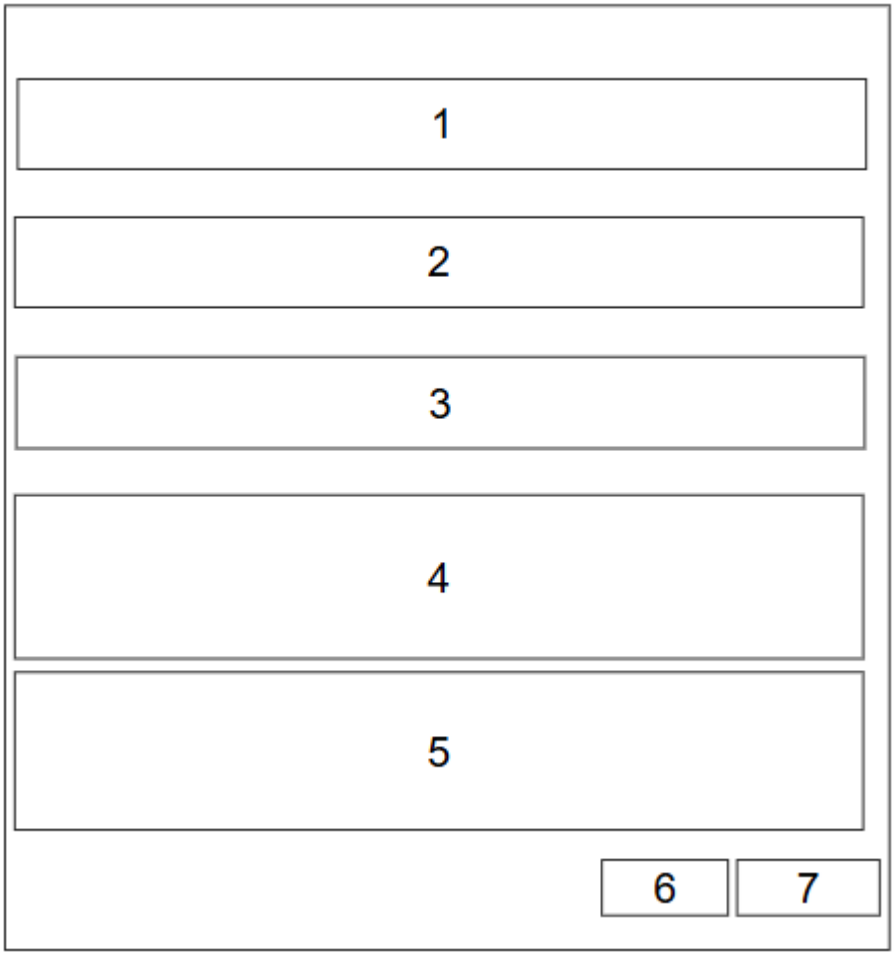


Рисунок 3.12 – Структурна схема діалогового вікна детального аналізу самопочуття

Складові елементи діалогового вікна детального аналізу самопочуття:

1. Рівень енергії.
2. Рівень стресу.
3. Тривалість сну.
4. Симптоми.
5. Додаткові нотатки.
6. Кнопка «Скасувати».
7. Кнопка «Зберегти».

Таким чином, розроблений інтерфейс забезпечує весь необхідний функціонал для для моніторингу біологічних показників людини і підтримки здорового способу життя. Він дозволяє відстежувати спожиті калорії та інші нутрієнти, обсяги

споживої води, тренування, надає рекомендації по тренуванням, раціону харчування, розпізнає кількість калорій та цінних речовин зі споживої їжі по фото.

3.4 Висновки

У третьому розділі було проведено аналіз мов програмування для розробки вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту, було обрано мову програмування TypeScript. Розроблено базу даних вебсервісу, сформовано схему бази даних, описано таблиці розробленої бази даних, які дозволяють зберігати, обробляти та витягувати дані згідно потреб користувача та функціоналу роботи системи. Розроблено інтерфейс користувача, наведено структуру інтерфейсу усіх вікон розробленого вебсервісу. Розроблений інтерфейс дозволяє додавати записи про спожиту їжу, воду, вести облік зміни ваги та отримувати інформацію про нутрієнтну цінність споживої їжі.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз методів тестування

Тестування вебсервісу для аналізу біологічних показників людини з використанням штучного інтелекту вимагає підходу, що охоплює різноманітні аспекти функціонування системи [22]. Враховуючи високу відповідальність за точність обробки даних, пов'язаних зі здоров'ям, необхідно застосувати структурований та багаторівневий процес забезпечення якості програмного продукту.

Функціональне тестування є початковим та фундаментальним етапом валідації розробленого вебсервісу. Під час цього виду тестування перевіряються основні операції, зокрема завантаження і розпізнавання зображень, аналіз нутрієнтного складу та збереження даних. Необхідно забезпечити підтримку різних форматів файлів (JPEG, PNG, HEIF), оцінити здатність системи обробляти фотографії різної якості та освітлення, а також підтвердити коректність ідентифікації різнотипних страв. Перевірка функціональності аналізу нутрієнтів вимагає порівняння результатів з еталонними даними про харчовий склад, оцінки точності алгоритмів перерахунку при різних вагових пропорціях та перевірки повноти отриманої інформації про макро- та мікроелементи.

Інтеграційне тестування становить наступний рівень забезпечення якості, спрямований на валідацію взаємодії між компонентами системи [23]. Цей процес включає перевірку комунікації між клієнтською та серверною частинами застосунку, оцінку правильності формування запитів до Claude Vision API та обробки отриманих відповідей, а також тестування операцій доступу до бази даних, включаючи транзакції та забезпечення цілісності інформації.

Безпека даних користувачів є пріоритетним аспектом у контексті сервісів, пов'язаних зі здоров'ям, що зумовлює необхідність комплексного тестування захисних механізмів [24]. Цей процес включає перевірку систем автентифікації та авторизації, оцінку захищеності маршрутів API та безпеки користувацьких сесій. Важливим елементом є тестування захисту даних, під час якого перевіряються

алгоритму шифрування при зберіганні та передачі інформації, верифікацію механізмів запобігання витоку конфіденційних даних та перевірку процедур анонімізації інформації для аналітичних цілей. Додатково проводиться оцінка стійкості системи до поширених вебвразливостей, таких як SQL-ін'єкції, XSS-атаки та CSRF-вразливості.

Тестування користувацького інтерфейсу є невід'ємною складовою забезпечення позитивного досвіду взаємодії з вебсервісом. Даний процес охоплює перевірку коректності відображення елементів інтерфейсу на різних пристроях, валідацію адаптивного дизайну та кросбраузерної сумісності. Юзабіліті-тестування оцінює інтуїтивність інтерфейсу, аналіз зручності виконання ключових сценаріїв використання та перевірку доступності для користувачів з обмеженими можливостями. Для оптимізації інтерфейсу застосовується A/B-тестування, що дозволяє порівняти різні варіанти дизайну та кількісно оцінити взаємодію користувачів з елементами інтерфейсу.

Регулярне застосування цих методів дозволяє досягти високої якості програмного продукту, що є важливим для системи моніторингу біологічних показників людини.

4.2 Тестування вебсервісу

Тестування розробленого вебсервісу починається з вітальної сторінки, у якій необхідно ввести своє ім’я (рисунок 4.1).

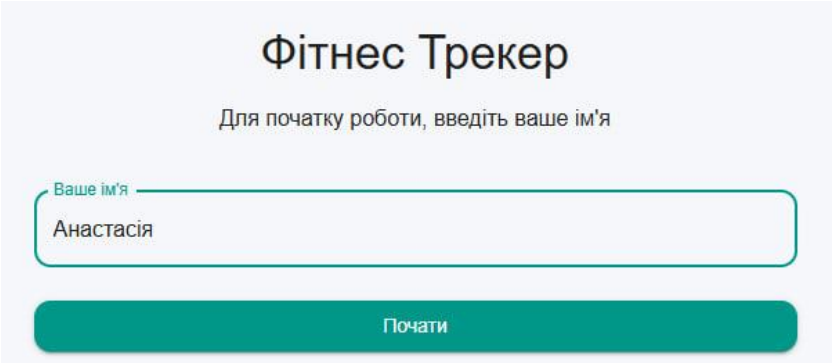


Рисунок 4.1 – Введення імені при запуску застосунку

Після цього запускається головна сторінка вебсервісу, яка містить 4 вкладки: «Вага», «Калорії», «Вода», «Тренування», при цьому першою за замовчуванням відображається вкладка «Вага». Вона дозволяє додавати записи ваги з метою моніторингу її зміни, та дивитися візуалізацію зміни ваги на графіку у правій частині вікна (рисунок 4.2).

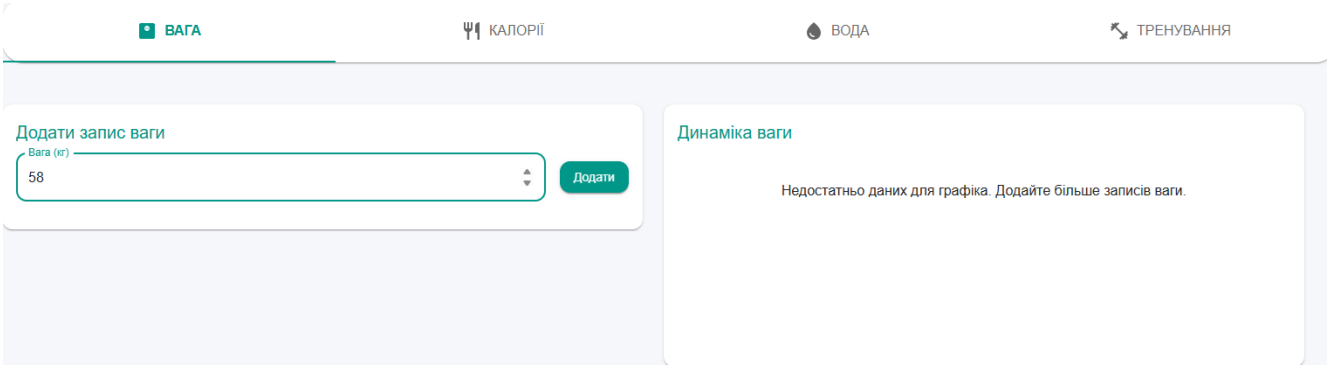


Рисунок 4.2 – Введення ваги для моніторингу

Побудований графік зміни ваги після доданих кількох записів наведено на рисунку 4.3.

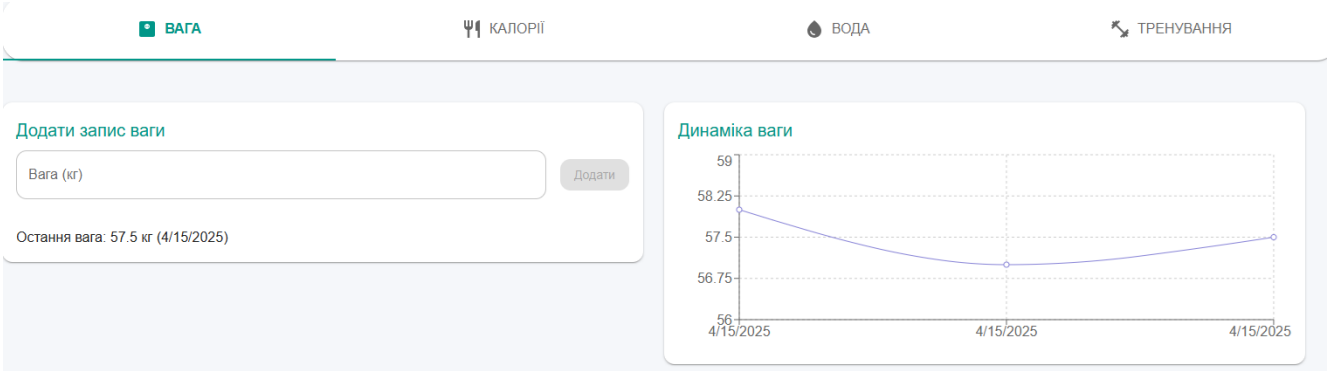


Рисунок 4.3 – Вкладка «Вага» та графік зміни ваги після заповнення даними

Вкладка «Калорії» відображає кількість спожитих калорій за день, прийоми їжі, та дозволяє додавати інформацію про прийоми їжі за день. Існує можливість як ручного додавання, так і розпізнавання за фото. Початковий стан цієї вкладки наведено на рисунку 4.4.

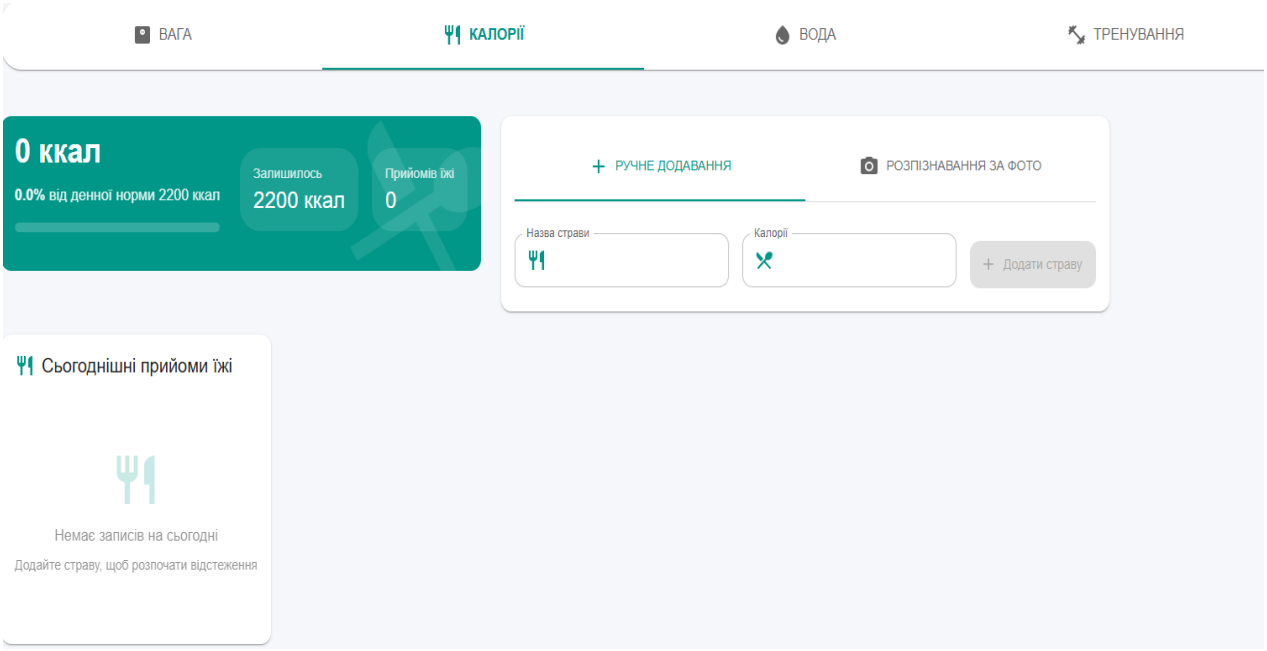


Рисунок 4.4 – Початковий стан вкладки «Калорії»

Введемо з використанням ручного додавання страву «Борщ український» та кількість калорій 450. Результат наведено на рисунку 4.5.

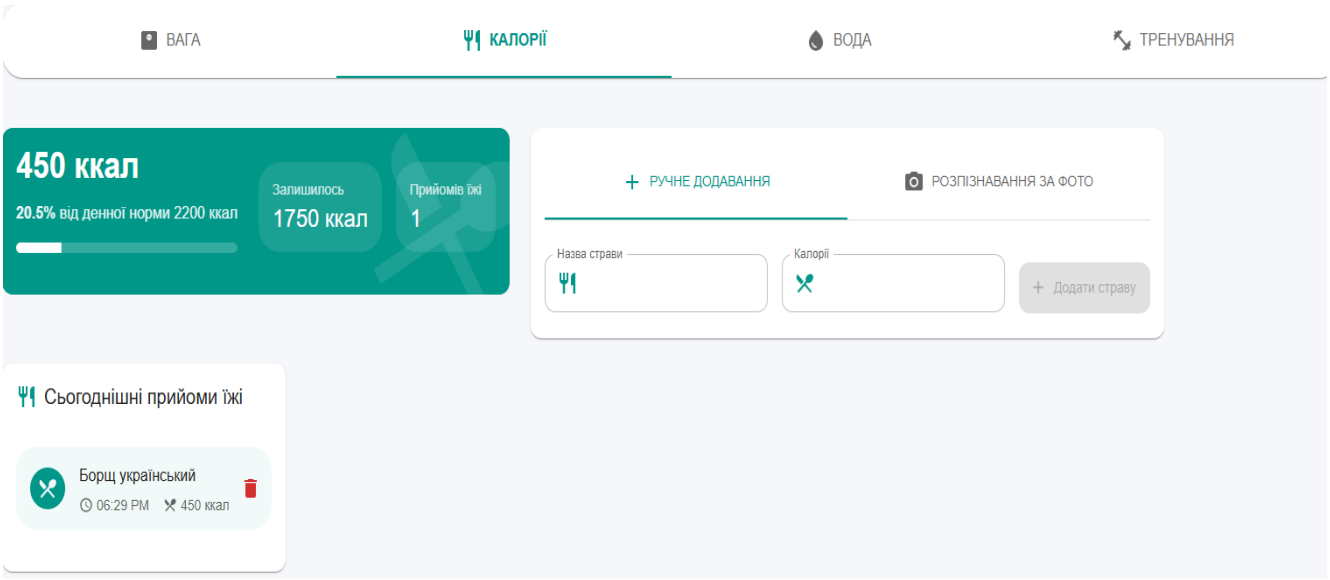


Рисунок 4.5 – Вкладка «Калорії» після введення даних вручну

На рисунку 4.6 наведено вкладку розпізнавання їжі за фото. Вона дозволяє додавати фото спожитої їжі, аналізувати, що саме на фото, та отримати інформацію про її нутрієнтну цінність у вигляді таблиці.

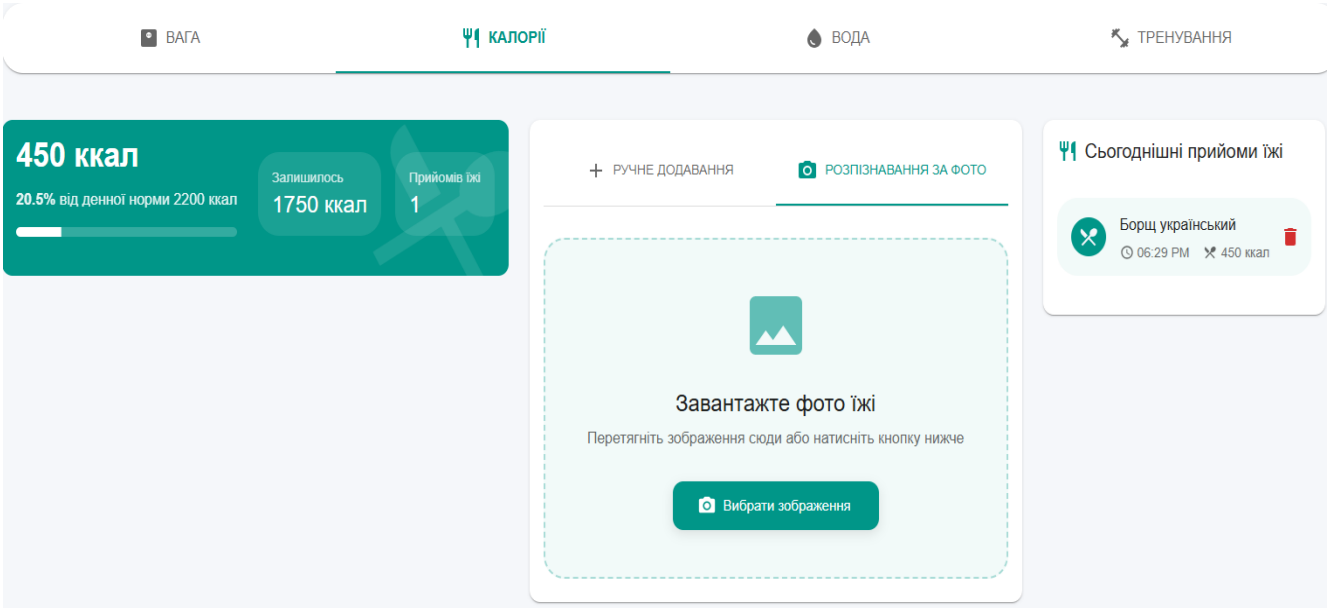


Рисунок 4.6 – Вкладка «Калорії» при виборі меню «Розпізнавання за фото»

Завантажимо фото суші Філадельфія, після чого розпочинається процес аналізу зображення, як на рисунку 4.7.

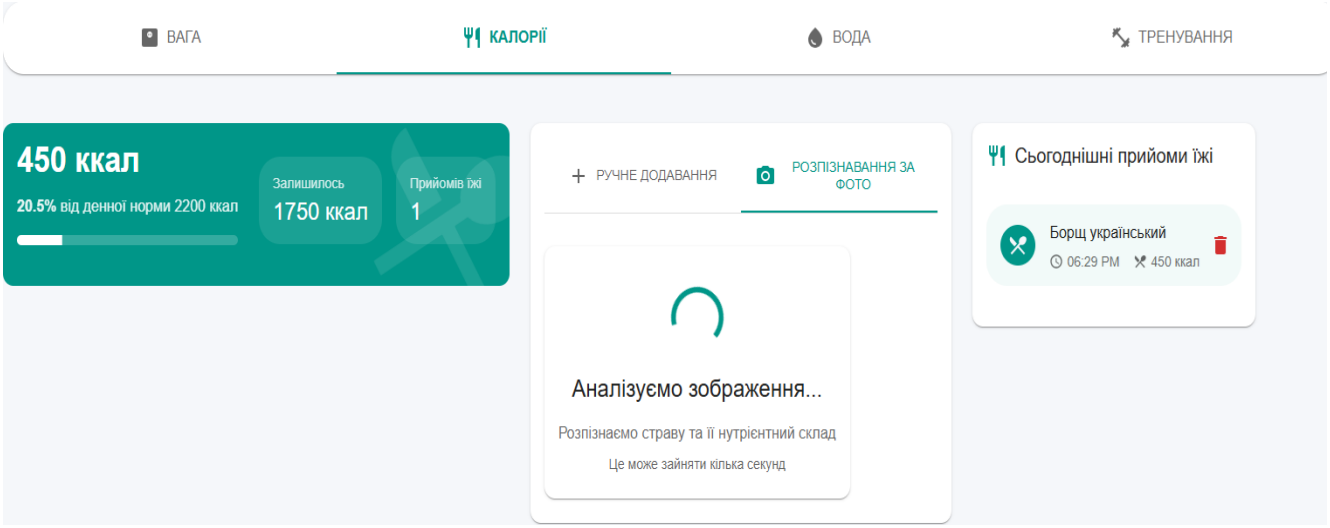


Рисунок 4.7 – Процес аналізу їжі за фото

На рисунку 4.8 продемонстровано результат аналізу зображення. Наведена інформативна таблиця з інформацією про поживні речовини та калорії завантаженої страви. Також можна перераховувати кількість цих речовин залежно від ваги реальної порції, адже за замовчуванням вказується 100г.

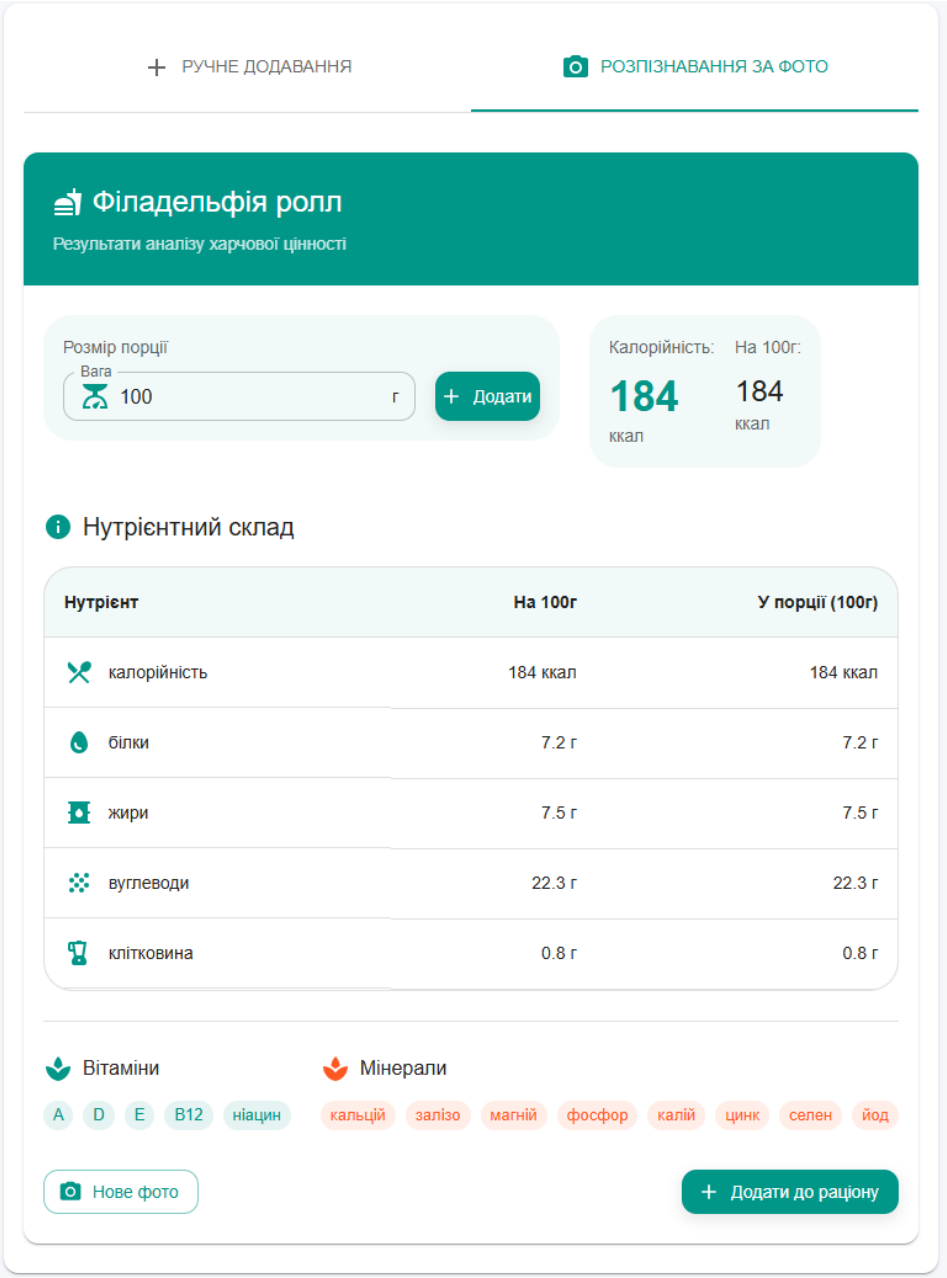


Рисунок 4.8 – Результат аналізу за фото

Змінимо вагу порції на 400г і збережемо отримані дані. Сповіщення про успішне збереження наведено на рисунку 4.9.

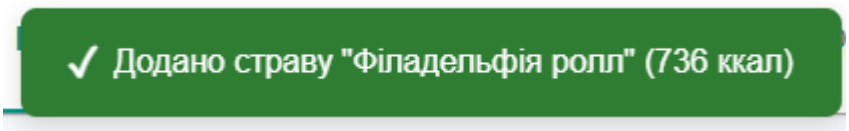


Рисунок 4.9 – Повідомлення про додавання страви до спожитих з перерахованою порцією їжі

На рисунку 4.10 наведено оновлений список прийомів їжі та кількість спожитих калорій.

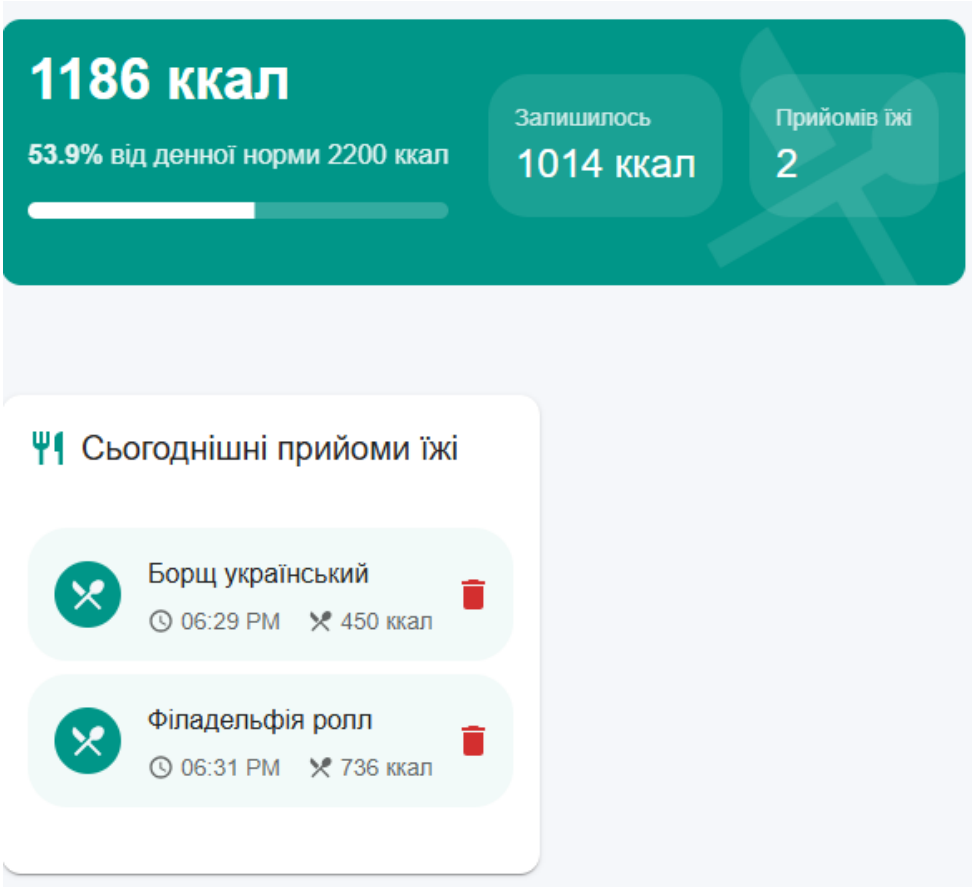


Рисунок 4.10 – Оновлена вкладка зі спожитими калоріями

На рисунку 4.11 наведена вкладка «Вода». Вона дозволяє робити записи про спожиту воду користувачем та звіряти виконання поставленого плану по спожитій воді за день.

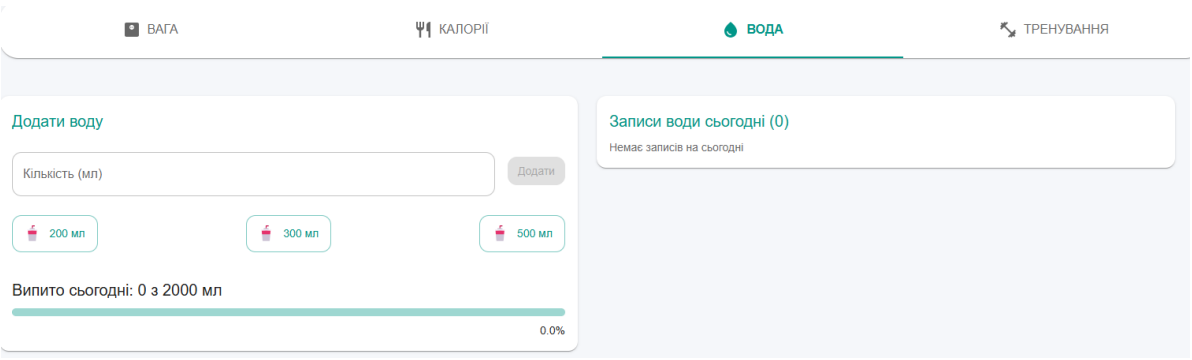


Рисунок 4.11 – Вкладка «Вода»

Додамо записи спожитої води за допомогою спеціальних кнопок, які додають фіксований обсяг спожитої води. Результат наведено на рисунку 4.12.

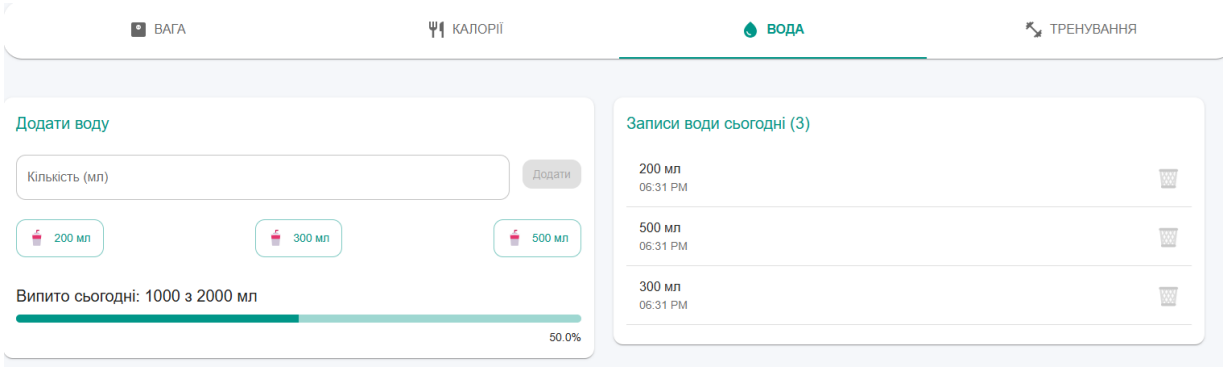


Рисунок 4.12 – Вкладка «Вода» після внесення даних

Перейдемо до вкладки тренувань. Тут містяться рекомендації щодо тренувань залежно від цільових показників ваги, поточної ваги, та спожитих калорій. Приклад рекомендації наведено на рисунку 4.13.

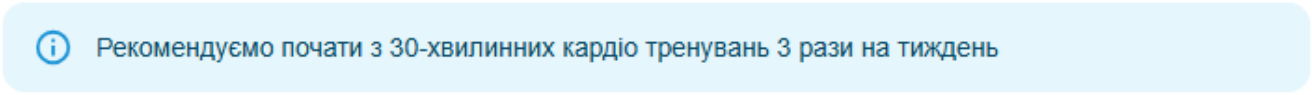


Рисунок 4.13 – Рекомендації щодо тренувань

На рисунку 4.14 продемонстровано процес вибору типу тренування. Він організований у вигляді випадного списку з різними видами тренувань.

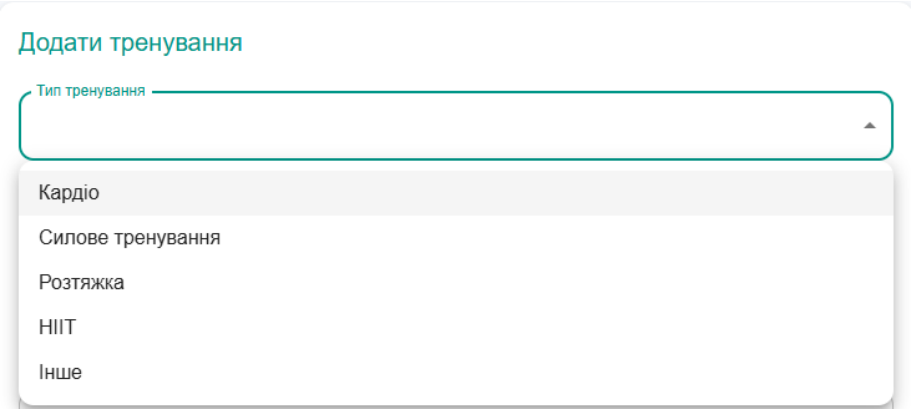


Рисунок 4.14 – Вибір тренувань

На рисунку 4.15 наведено додане тренування у списку доданих тренувань.

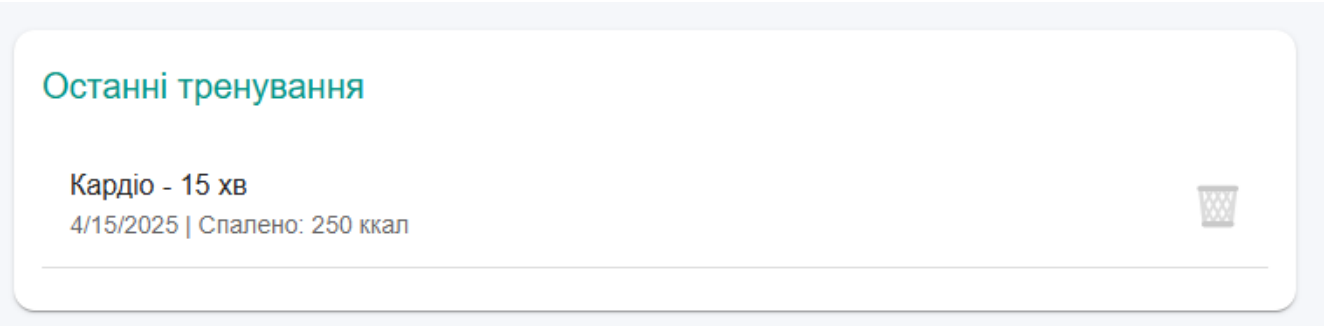


Рисунок 4.15 – Вкладка «Останні тренування» після доданого тренування

Таким чином, проведене тестування демонструє, що розроблений вебсервіс виконує поставлені завдання, працює відповідно до очікувань, дозволяє контролювати біологічні показники людини.

На рисунку 4.16 показано сторінку для оцінки самопочуття користувача. Система пропонує п'ять варіантів настрою: "Дуже погано", "Погано", "Нормально", "Добре", "Відмінно". Користувач може обрати відповідний варіант, та додати детальний запис через кнопку "Додати детальний запис".

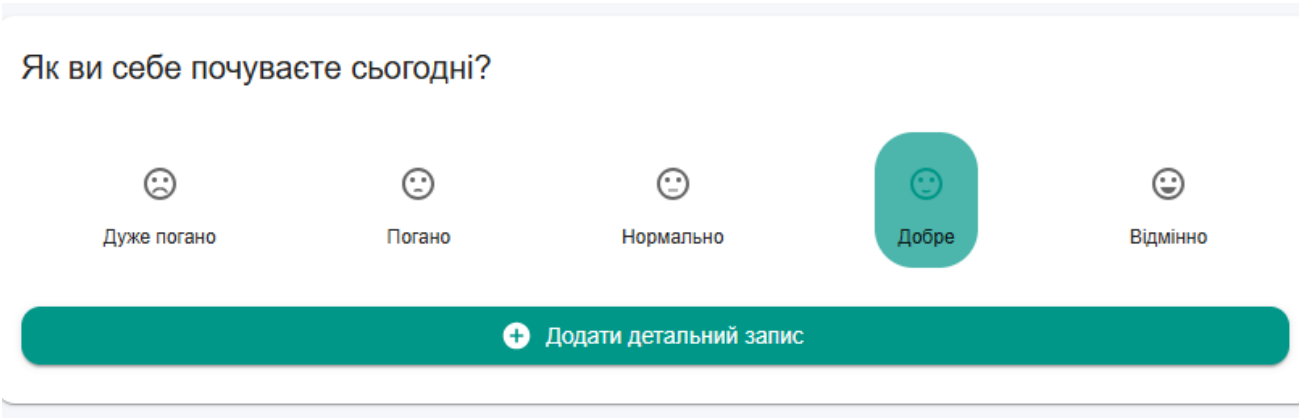


Рисунок 4.16 – Сторінка для оцінювання самопочуття користувача

На рисунку 4.17 демонструється форма детального запису самопочуття. Система дозволяє оцінити рівень енергії, рівень стресу та тривалість сну за допомогою слайдерів. Також є можливість вибрати симптоми з переліку: головний біль, втома, тривожність, роздратованість, концентрація, апетит, безсоння, нудота. Користувач може додати додаткові нотатки у відповідному полі.

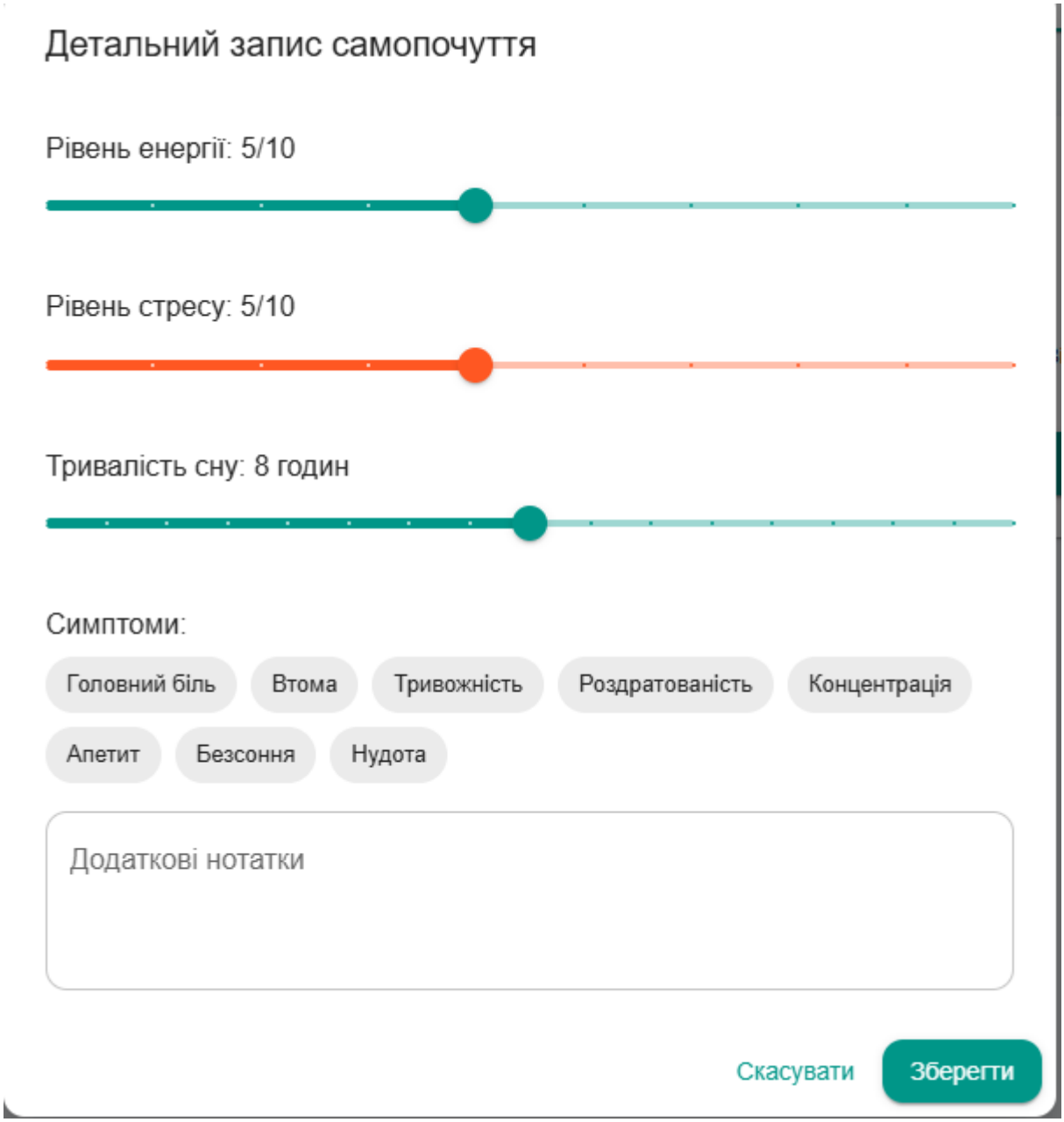


Рисунок 4.17 – Діалог детального запису самопочуття

На рисунку 4.18 продемонстровано сторінку з відображенням статистики за тиждень після внесення даних. Ця сторінка відображає інформацію про середній настрій, середню енергію та середній стрес у вигляді прогрес-барів. У розділі «Персональні рекомендації» система надає поради. В розділі «Останні записи» відображаються останні додані записи настрою.

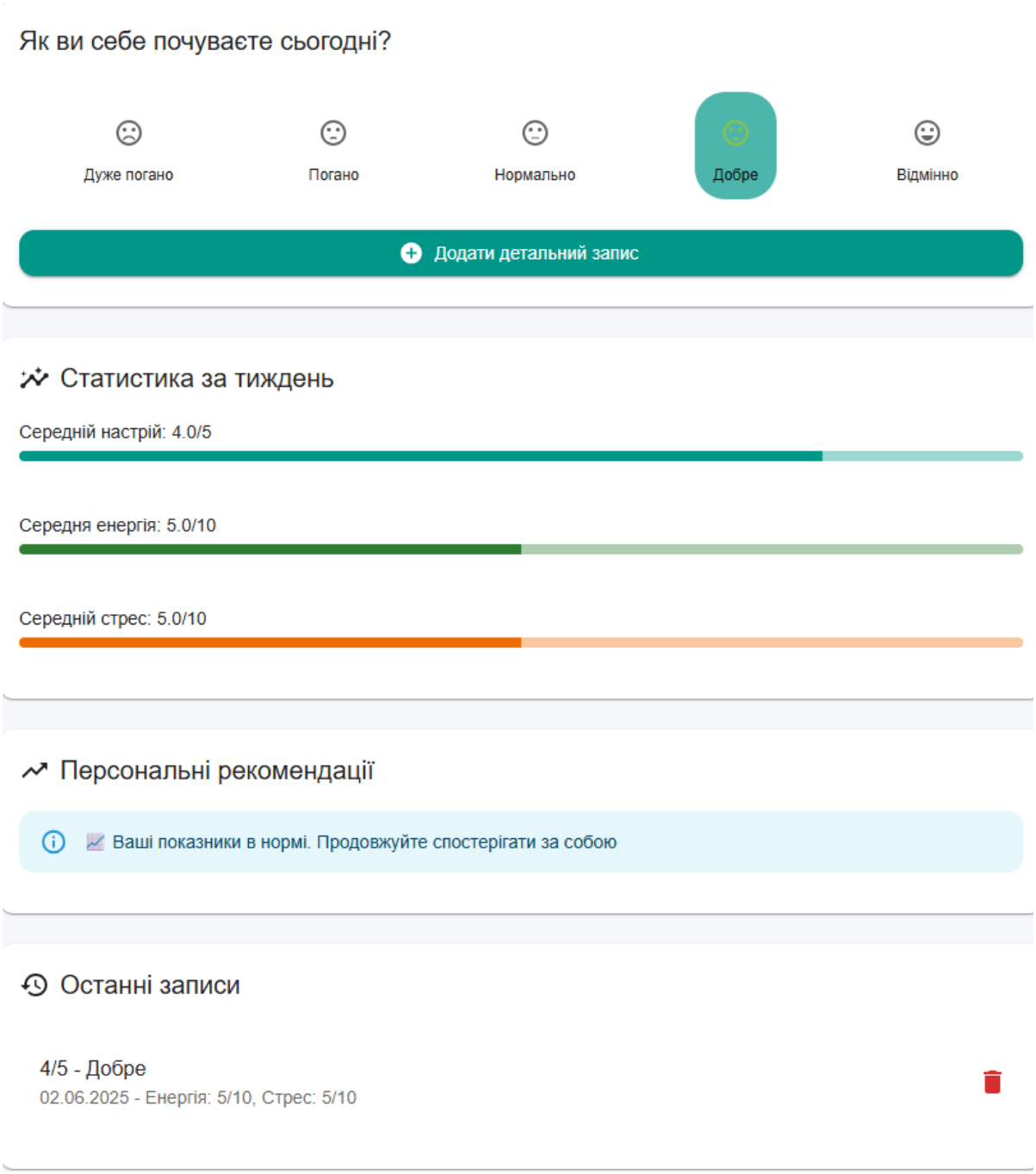


Рисунок 4.18 – Сторінка з інформацією про настрій зі статистикою та останніми записами

На рисунку 4.19 наведено інтерфейс для додавання записів про сон. Користувач може вказати час відходу до сну та час прокидання, після чого система автоматично розраховує тривалість сну. Також передбачена можливість оцінити якість сну за допомогою п'ятизіркової шкали та зберегти дані через кнопку «Зберегти дані про сон».

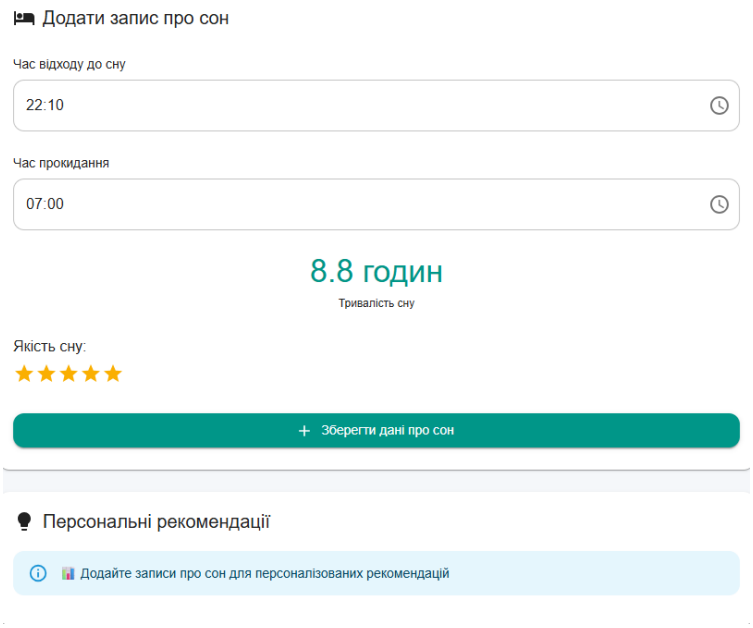


Рисунок 4.19 – Додавання інформації про сон

На рисунку 4.20 продемонстровано статистику сну після внесення даних. Відображається середня тривалість за тиждень та середня якість сну.

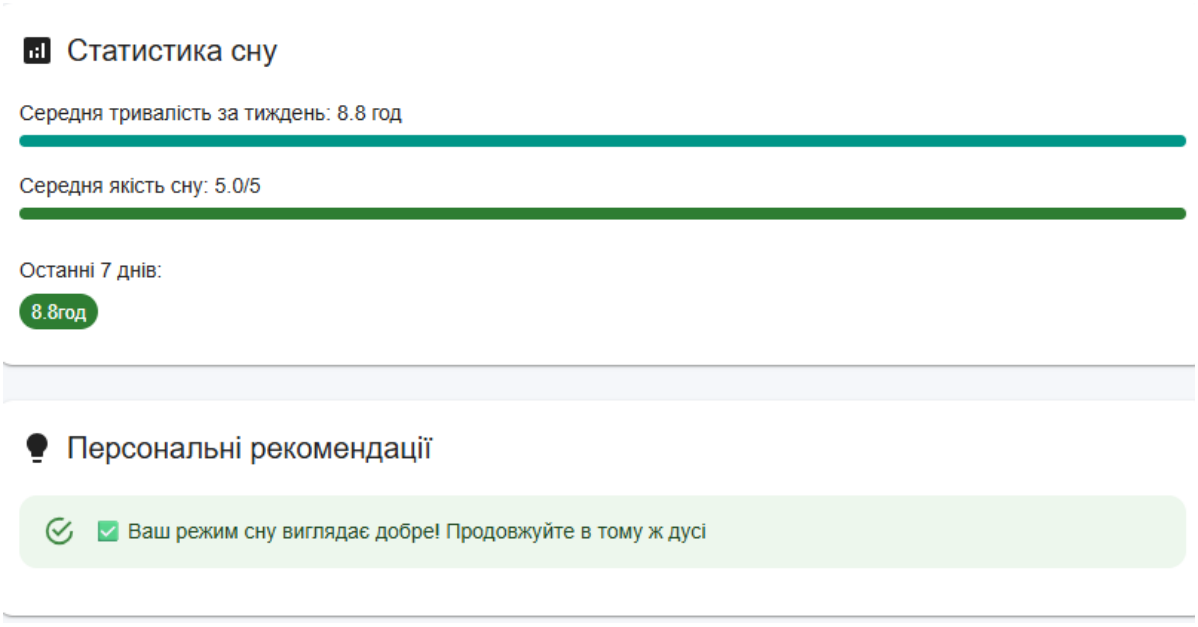


Рисунок 4.20 – Статистика сну

На рисунку 4.21 показано сторінку рекомендацій. Кожна рекомендація має кнопку «Позначити як виконано» для позначення її виконання.

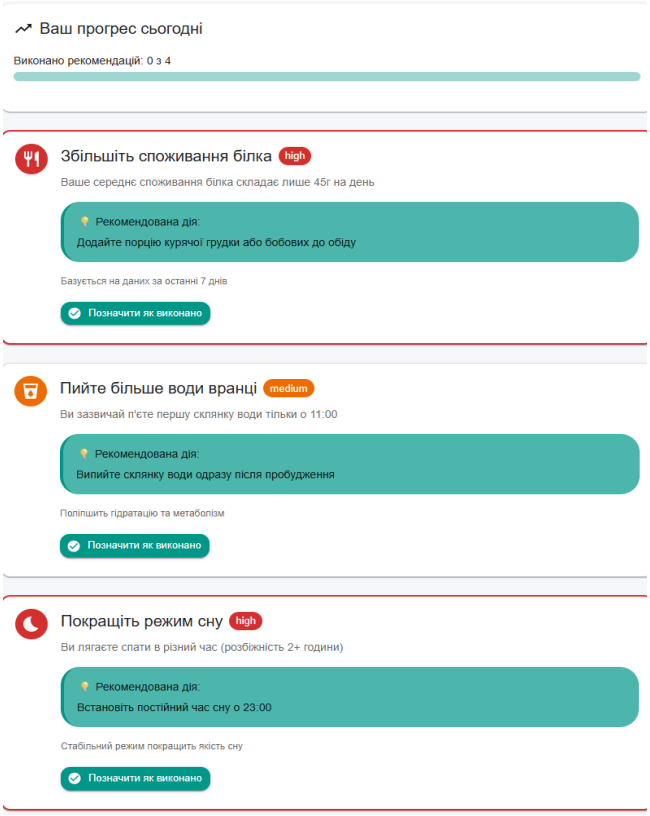


Рисунок 4.21 – Сторінка рекомендацій

Позначимо першу рекомендацію як таку, що виконану. Ну рисунку 4.22 продемонстровано оновлену панель прогресу (1 з 4 виконано) після відмітки першої рекомендації як виконаної. Рекомендація про споживання білка тепер має зелену позначку «Виконано».

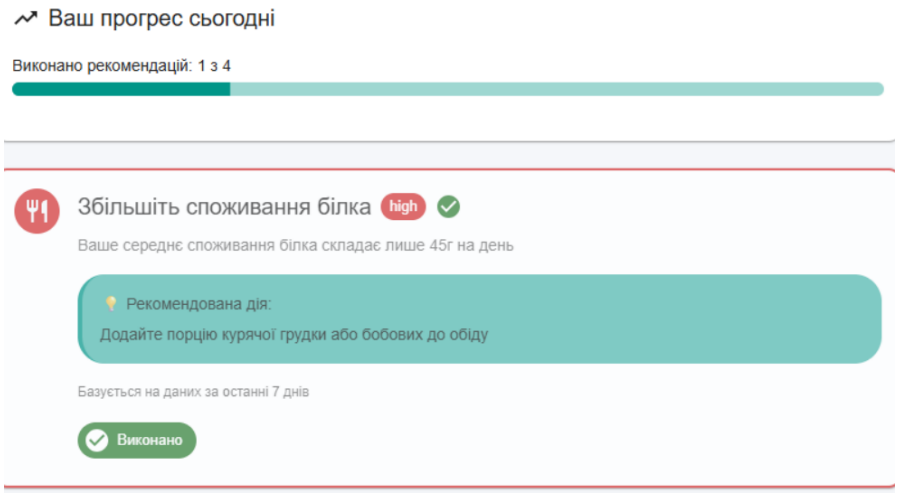


Рисунок 4.22 – Оновлена сторінка рекомендацій

На рисунку 4.23 наведено детальну панель статистики за тиждень. Система

відображає чотири основні метрики: середні калорії, спожита вода щодня, тренування, середній сон. У розділі «Детальний аналіз» система надає прогрес досягнення цілей, тренди, рекомендації.

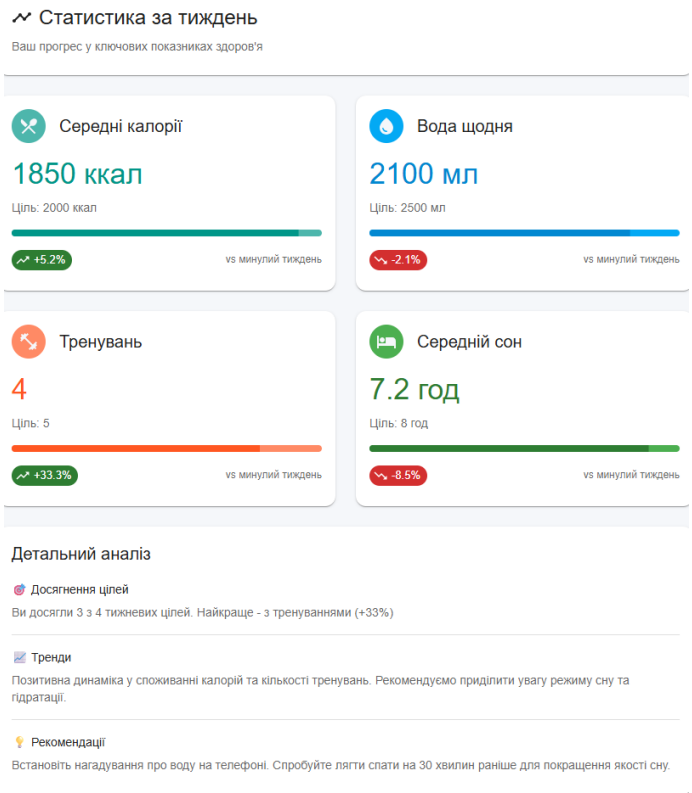


Рисунок 4.23 – Панель статистики за тиждень

Таким чином, проведене тестування демонструє, що розроблений вебсервіс виконує поставлені завдання, працює відповідно до очікувань, дозволяє контролювати біологічні показники людини.

4.3 Висновки

У четвертому розділі було проведено аналіз методів тестування вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту. Проведено тестування розробленого сервісу. Було продемонстровано його можливості, наведено приклад використання, продемонстровано, що розроблений застосунок виконує поставлені на початку розробки завдання.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи, було розроблено вебсервіс для контролю біологічних показників людини з використанням штучного інтелекту. Розроблений сервіс дозволяє відстежувати зміну ваги користувача, проводити аналіз спожитої їжі, випитої води, отримувати рекомендації щодо тренування та харчування.

Було проведено аналіз сучасного стану застосунків для контролю біологічних показників людини, доведено актуальність програмної розробки в цьому напрямку, проведено порівняльний аналіз аналогів.

Для розробки було обрано мову програмування TypeScript та середовище розробки Visual Studio Code, Claude Vision API для розпізнавання їжі за фото, Node.js для серверної частини застосунку, React.js для інтерфейсу користувача. Бакалаврську кваліфікаційну роботу реалізовано згідно методичних вказівок [25].

У першому розділі було проведено аналіз сучасного стану застосунків для контролю біологічних показників людини, доведено актуальність програмної розробки в цьому напрямку, проведено порівняльний аналіз аналогів: Samsung Health, MyFitnessPal, Yazio, Cronometer, продемонстровано переваги власної розробки перед аналогами, проведено аналіз методів розв’язання задачі, поставлено задачі дослідження на розробку вебзастосунку для контролю біологічних показників людини з використанням штучного інтелекту.

У другому розділі було проведено аналіз інформаційного забезпечення вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту, визначено дані, які збираються, обробляються, генеруються при роботі вебсервісу. Розроблено модель системи у три шари – клієнтська частина, серверна та мережева. Описано компоненти, що використовуються у роботі вебсервісу. Розроблено архітектуру вебсервісу, побудовано деревовидну структуру, що містить усі класи, методи, таблиці для роботи вебсервісу. Розроблено методи автоматичного аналізу харчування на основі комп’ютерного зору та персоналізованих адаптивних рекомендацій.

У третьому розділі було проведено аналіз мов програмування для розробки вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту, було обрано мову програмування TypeScript. Розроблено базу даних вебсервісу, сформовано схему бази даних, описано таблиці розробленої бази даних, які дозволяють зберігати, обробляти та витягувати дані згідно потреб користувача та функціоналу роботи системи. Розроблено інтерфейс користувача, наведено структуру інтерфейсу усіх вікон розробленого вебсервісу. Розроблений інтерфейс дозволяє додавати записи про спожиту їжу, воду, вести облік зміни ваги та отримувати інформацію про нутрієнтну цінність спожитої їжі.

У четвертому розділі було проведено аналіз методів тестування вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту. Проведено тестування розробленого сервісу. Було продемонстровано його можливості, наведено приклад використання, продемонстровано, що розроблений застосунок виконує поставлені на початку розробки завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mobile Health Monitoring System: A Comprehensive Review [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/372000639_Mobile_

2. Health_Monitoring_System_A_Comprehensive_Review. Insights into mobile health application market via a content analysis of marketplace data with machine learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7787530/

3. Волос А.В. Аналіз мобільних додатків для відстеження біологічних показників здоров'я людини / Матеріали IV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів і студентів, Одеса, 26-27 вересня 2024 р. – Одеса, Видавництво ОНТУ, 2024 р. – С.121-122.

4. van den Boer R., Helms M., Ghosh S. Use of Different Food Image Recognition Platforms in Dietary Assessment. JMIR Formative Research, 2020;4(12):e15602.

5. Ciocca G., Baldini G., Caldarola G. et al. A review on food recognition technology for health applications. Computers in Biology and Medicine, PMC7859960, 2021.

6. Harvard T.H. Chan School of Public Health. Precision Nutrition.

7. Breysse P., Baccarelli A. Transforming the cardiometabolic disease landscape: Multimodal AI holds great promise. PNAS, PMC10990799, 2024.

8. Hariprasad C., Ventura J. et al. When and Why Adults Abandon Lifestyle Behavior and Mental Apps. J Med Internet Res, 2024;26:e56897.

9. National Academies of Sciences, Engineering, and Medicine. Advances in Multimodal Artificial Intelligence to Enhance Environmental and Biomedical Data Integration. Proceedings of a Workshop-in-Brief, 2023.

10. Samsung Health [Електронний ресурс] – Режим доступу до ресурсу: https://www.samsung.com/ua/apps/samsung-health/?srsltid=AfmBOoqdxln2FUbj_E3WSWunsvDszjOXwFyK--13iaHCksF3fCmy_5MI

11. MyFitnessPal [Електронний ресурс] – Режим доступу до ресурсу: https://www.myfitnesspal.com/

12. Yazio [Електронний ресурс] – Режим доступу до ресурсу: https://www.yazio.com/en

13. Cronometer [Електронний ресурс] – Режим доступу до ресурсу: https://cronometer.com/

14. Definition of monolithic architecure [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/monolithic-architecture

15. Microservices vs Monolith [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

16. Serverless [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/think/topics/serverless

17. Typescript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

18. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/

19. David Flanagan. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language 7th Edition. Farnham: O’Reilly Media, 2020. 704с.

20. A. Petrov. Database Internals: A Deep Dive into How Distributed Data Systems Work. O’Reilly Media, 2020. 370с.

21. What is UI Design? [Електронний ресурс] – Режим доступу до ресурсу: https://www.interaction-design.org/literature/topics/ui-design?srsltid=AfmBOorNV7d1KXCCxvvNWT9LEysi_BdZ8WtnugoxL6NMnSwYlwP_Q1xk

22. A. Samaru. Software Testing: An ISTQB-BCS Certified Tester Foundation Level guide (CTFL v4.0) – Fifth edition. Лондон: BCS, 2024. 283с.

23. A. Axelrod. Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects. Нью-Йорк: Apress, 2021. 558с.

24. A.Kovach. Modern Software Testing Techniques: A Practical Guide for Developers and Testers. Нью-Йорк: Apress, 2024. 266с.

25. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О.Н. Романюк, Г.О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«25» 03 2025 року

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Мельник О.В.
«24» 03 2025 року.

Виконала:
студентка гр. 2ПІ-21б Волос А.В.
«24» 03 2025 року.

м. Вінниця – 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту».

Галузь застосування – вебтехнології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 18 від 18 лютого 2025 р.

3. Мета та призначення розробки.

Метою роботи є підвищення рівня контролю біологічних показників людини шляхом розробки спеціалізованої системи, що дозволяє відслідковування та контроль біологічних показників людини.

4. Вихідні дані для проведення НДР

1. Mobile Health Monitoring System: A Comprehensive Review [Електроний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/372000639_Mobile_

2. Typescript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

3. A. Petrov. Database Internals: A Deep Dive into How Distributed Data Systems Work. O'Reilly Media, 2020. 370c.

4. What is UI Design? [Електронний ресурс] – Режим доступу до ресурсу: https://www.interaction-design.org/literature/topics/ui-design?srsltid=AfmBOorNV7d1KXCCxvvNWT9LEysi_BdZ8WtnugoxL6NMnSwYlwP_Q1

5. A. Samaru. Software Testing: An ISTQB-BCS Certified Tester Foundation Level guide (CTFL v4.0) – Fifth edition. Лондон: BCS, 2024. 283c.

5. Технічні вимоги

Вхідні дані – ім'я користувача, вага, цільові показники калорій та ваги.

Вихідні дані – графіки зі статистикою, аналіз спожитих страв.

6. Конструктивні вимоги.

Інтерфейс програми поовинен відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

–пояснювальна записка до бакалаврської кваліфікаційної роботи;

–технічне завдання;

–лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку вебсервісів для контролю біологічних показників людини з використанням штучного інтелекту	25.03.25 – 05.04.25
2	Розробка структури та алгоритмів програмного застосунку	06.04.25 – 18.04.25
3	Варіантний аналіз та вибір мови програмування розробки	19.04.25 – 21.04.25
4	Розробка системи для моніторингу біологічних показників людини і підтримки способу життя	22.04.25 – 17.05.25
5	Тестування програмного забезпечення	18.05.25 – 25.05.25
6	Оформлення матеріалів до захисту БКР	26.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Усі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота

Підрозділ кафедра ПЗ, ФІТКІ, 2ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 3,6 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Мельник О. В., к. т. н., доцент кафедри ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Волос А. В.
(підпис) (прізвище, ініціали)

Додаток В – Лістинг програми

```
import React, { useState } from 'react';
import {
  Grid, Typography, TextField, Button, Box, List, ListItem, ListItemText,
  ListItemSecondaryAction, IconButton, Divider, LinearProgress, Tab, Tabs, Card,
  CardContent, alpha, useTheme, Avatar
} from '@mui/material';

import { HealthTracker } from './services/HealthTracker';
import { Meal } from './models/types';
import FoodAnalyzer from './FoodAnalyzer';

// Імпортуйте іконки
import DeleteIcon from '@mui/icons-material/Delete';
import AddIcon from '@mui/icons-material/Add';
import RestaurantIcon from '@mui/icons-material/Restaurant';
import PhotoCameraIcon from '@mui/icons-material/PhotoCamera';
import AccessTimeIcon from '@mui/icons-material/AccessTime';
import LocalDiningIcon from '@mui/icons-material/LocalDining';
import TrendingUpIcon from '@mui/icons-material/TrendingUp';
import TrendingDownIcon from '@mui/icons-material/TrendingDown';

interface CalorieTrackerProps {
  tracker: HealthTracker;
}

interface TabPanelProps {
  children?: React.ReactNode;
  index: number;
  value: number;
}

function TabPanel(props: TabPanelProps) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      {value === index && <Box sx={{ pt: 2 }}>{children}</Box>}
    </div>
  );
}

const CalorieTracker: React.FC<CalorieTrackerProps> = ({ tracker }) => {
```

```
const theme = useTheme();
const [mealName, setMealName] = useState('');
const [calories, setCalories] = useState('');
const [tabValue, setTabValue] = useState(0);
const [refreshKey, setRefreshKey] = useState(0);


const user = tracker.getUser();
const targetCalories = user.targetCalories || 2000;
const consumedCalories = tracker.getTodayCalories();
const remainingCalories = tracker.getRemainingCalories();
const meals = tracker.getTodayMeals();


const handleAddMeal = () => {
  if (mealName && calories && parseInt(calories) > 0) {
    tracker.addMeal(mealName, parseInt(calories));
    setMealName('');
    setCalories('');
    setRefreshKey(prev => prev + 1);
  }
};


const handleDeleteMeal = (id: string) => {
  tracker.deleteMeal(id);
  setRefreshKey(prev => prev + 1);
};


const formatTime = (date: Date) => {
  return new Date(date).toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' });
};


const calculateCaloriePercentage = () => {
  return Math.min(100, (consumedCalories / targetCalories) * 100);
};

const handleTabChange = (event: React.SyntheticEvent, newValue: number) => {
  setTabValue(newValue);
};

return (
  <Grid container spacing={3}>
    {/* Статистика калорій */}
    <Grid item component="div" xs={12}>
      <Card elevation={0} sx={{
        bgcolor: consumedCalories > targetCalories ? 'error.main' : 'primary.main',
        color: 'white',
        position: 'relative',
        overflow: 'hidden'
      }}>
```

```
<Box
  sx={{
    position: 'absolute',
    top: -30,
    right: -20,
    fontSize: 200,
    opacity: 0.1,
    transform: 'rotate(15deg)',
    color: 'white'
  }}
>
  <LocalDiningIcon sx={{ fontSize: 'inherit' }} />
</Box>
<CardContent sx={{ position: 'relative' }}>
  <Grid container spacing={3} alignItems="center">
    <Grid item xs={12} md={6}>
      <Typography variant="h4" fontWeight="bold" gutterBottom>
        {consumedCalories} ккал
      </Typography>
      <Typography variant="body1" sx={{ opacity: 0.9, mb: 2 }}>
        <strong>{calculateCaloriePercentage().toFixed(1)}%</strong> від денної норми {targetCalories} ккал
      </Typography>
      <LinearProgress
        variant="determinate"
        value={calculateCaloriePercentage()}
        sx={{
          height: 10,
          borderRadius: 5,
          mb: 2,
          bgcolor: alpha('#fff', 0.2),
          '& .MuiLinearProgress-bar': {
            bgcolor: 'white'
          }
        }}
      />
    </Grid>
    <Grid item xs={12} md={6}>
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <Box sx={{ bgcolor: alpha('#fff', 0.1), p: 2, borderRadius: 2 }}>
            <Typography variant="subtitle2" sx={{ opacity: 0.8 }}>
              Залишилось
            </Typography>
            <Typography variant="h5" fontWeight="medium">
              {remainingCalories > 0 ? remainingCalories : 0} ккал
            </Typography>
          </Box>
        </Grid>
        <Grid item xs={6}>
          <Box sx={{ bgcolor: alpha('#fff', 0.1), p: 2, borderRadius: 2 }}>
            <Typography variant="subtitle2" sx={{ opacity: 0.8 }}>
              Прийомів їжі
```

```
            </Typography>
            <Typography variant="h5" fontWeight="medium">
              {meals.length}
            </Typography>
          </Box>
        </Grid>
      </Grid>
    </Grid>
  </CardContent>
 </Card>
</Grid>

{/* Додавання їжі */}
<Grid item xs={12}>
 <Card>
  <CardContent>
   <Tabs
    value={tabValue}
    onChange={handleTabChange}
    variant="fullWidth"
    indicatorColor="primary"
    textColor="primary"
    sx={{ mb: 2, borderBottom: 1, borderColor: 'divider' }}
   >
    <Tab icon={<AddIcon />} label="Ручне додавання" iconPosition="start" />
    <Tab icon={<PhotoCameraIcon />} label="Розпізнавання за фото" iconPosition="start" />
   </Tabs>


   <TabPanel value={tabValue} index={0}>
    <Grid container spacing={2}>
     <Grid item xs={12} md={8}>
      <TextField
       label="Назва страви"
       value={mealName}
       onChange={(e) => setMealName(e.target.value)}
       fullWidth
       variant="outlined"
       InputProps={{
        startAdornment: (
         <RestaurantIcon sx={{ color: 'primary.main', mr: 1 }} />
        ),
       }}
      />
     </Grid>
     <Grid item xs={12} md={4}>
      <TextField
       label="Калорії"
       type="number"
       value={calories}
       onChange={(e) => setCalories(e.target.value)}
```

```
            fullWidth
            variant="outlined"
            InputProps={{
              startAdornment: (
                <LocalDiningIcon sx={{ color: 'primary.main', mr: 1 }} />
              ),
            }}
          />
        </Grid>
        <Grid item xs={12}>
          <Button
            variant="contained"
            color="primary"
            onClick={handleAddMeal}
            disabled={!mealName || !calories}
            startIcon={<AddIcon />}
            fullWidth
            sx={{ mt: 1, py: 1.5 }}
          >
            Додати страву
          </Button>
        </Grid>
      </Grid>
    </TabPanel>


    <TabPanel value={tabValue} index={1}>
      <FoodAnalyzer tracker={tracker} />
    </TabPanel>
  </CardContent>
 </Card>
</Grid>


{/* Історія прийомів їжі */}
<Grid item xs={12}>
 <Card>
  <CardContent>
    <Typography variant="h6" sx={{ mb: 3, display: 'flex', alignItems: 'center' }}>
      <RestaurantIcon sx={{ mr: 1, color: 'primary.main' }} />
      Сьогоднішні прийоми їжі
    </Typography>


    {meals.length > 0 ? (
      <List sx={{ px: 0 }}>
        {meals.map((meal: Meal) => (
          <ListItem
            key={meal.id}
            sx={{
              mb: 1,
              bgcolor: alpha(theme.palette.primary.main, 0.05),
              borderRadius: 2,
```

```
        '&:hover': {
          bgcolor: alpha(theme.palette.primary.main, 0.1),
        }
      }}
    >
      <Avatar sx={{ bgcolor: 'primary.main', mr: 2 }}>
        <LocalDiningIcon />
      </Avatar>
      <ListItemText
        primary={<Typography variant="subtitle1" fontWeight="medium">{meal.name}</Typography>}
        secondary={
          <Box sx={{ display: 'flex', alignItems: 'center', mt: 0.5 }}>
            <AccessTimeIcon sx={{ fontSize: 16, mr: 0.5, color: 'text.secondary' }} />
            <Typography variant="body2" color="text.secondary" sx={{ mr: 2 }}>
              {formatTime(meal.time)}
            </Typography>
            <LocalDiningIcon sx={{ fontSize: 16, mr: 0.5, color: 'text.secondary' }} />
            <Typography variant="body2" color="text.secondary">
              {meal.calories} ккал
            </Typography>
          </Box>
        }
      />
      <ListItemSecondaryAction>
        <IconButton
          edge="end"
          onClick={() => handleDeleteMeal(meal.id)}
          sx={{
            color: 'error.main',
            '&:hover': {
              bgcolor: alpha(theme.palette.error.main, 0.1),
            }
          }}
        >
          <DeleteIcon />
        </IconButton>
      </ListItemSecondaryAction>
    </ListItem>
  ))}
 </List>
) : (
 <Box
   sx={{
     display: 'flex',
     flexDirection: 'column',
     alignItems: 'center',
     py: 6,
     opacity: 0.7
   }}
 >
   <RestaurantIcon sx={{ fontSize: 60, color: 'primary.main', opacity: 0.3, mb: 2 }} />
   <Typography variant="body1" color="text.secondary">
```

```
              Немає записів на сьогодні
            </Typography>
            <Typography variant="body2" color="text.secondary" sx={{ mt: 1 }}>
              Додайте страву, щоб розпочати відстеження
            </Typography>
          </Box>
        )}
      </CardContent>
    </Card>
  </Grid>
 </Grid>
 );
};

export default CalorieTracker;

import React, { useState, useRef } from 'react';
import {
  Card, CardContent, Typography, Box, Button, CircularProgress,
  Table, TableBody, TableCell, TableContainer, TableHead, TableRow,
  TextField, Grid, Alert, alpha, useTheme, Divider, Chip, Paper
} from '@mui/material';
import { HealthTracker } from './services/HealthTracker';

// Імпортуйте іконки
import PhotoCameraIcon from '@mui/icons-material/PhotoCamera';
import AddIcon from '@mui/icons-material/Add';
import FoodBankIcon from '@mui/icons-material/FoodBank';
import LocalDiningIcon from '@mui/icons-material/LocalDining';
import InfoIcon from '@mui/icons-material/Info';
import ImageIcon from '@mui/icons-material/Image';
import FastfoodIcon from '@mui/icons-material/Fastfood';
import ScaleIcon from '@mui/icons-material/Scale';
import EggIcon from '@mui/icons-material/Egg';
import OilBarrelIcon from '@mui/icons-material/OilBarrel';
import GrainIcon from '@mui/icons-material/Grain';
import BlenderIcon from '@mui/icons-material/Blender';
import VitaminIcon from '@mui/icons-material/Spa';

interface FoodAnalyzerProps {
  tracker: HealthTracker;
}

interface NutrientData {
  назва: string;
  калорійність: number;
  білки: number;
  жири: number;
  вуглеводи: number;
  клітковина?: number;
  вітаміни?: string[];
  мінерали?: string[];
```

```
  [key: string]: any;
}

const FoodAnalyzer: React.FC<FoodAnalyzerProps> = ({ tracker }) => {
  const theme = useTheme();
  const [image, setImage] = useState<string | null>(null);
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const [foodData, setFoodData] = useState<NutrientData | null>(null);
  const [portionSize, setPortionSize] = useState<string>('100');
  const fileInputRef = useRef<HTMLInputElement>(null);
  const dropAreaRef = useRef<HTMLDivElement>(null);

  const handleImageUpload = (e: React.ChangeEvent<HTMLInputElement>) => {
    if (e.target.files && e.target.files.length > 0) {
      const file = e.target.files[0];
      if (!file.type.match('image.*')) {
        setError('Будь ласка, завантажте зображення');
        return;
      }

      const reader = new FileReader();
      reader.onload = (event) => {
        if (event.target) {
          setImage(event.target.result as string);
        }
      };
      reader.readAsDataURL(file);

      analyzeFood(file);
    }
  };

  const handleDragOver = (e: React.DragEvent<HTMLDivElement>) => {
    e.preventDefault();
    e.stopPropagation();
    if (dropAreaRef.current) {
      dropAreaRef.current.style.borderColor = theme.palette.primary.main;
      dropAreaRef.current.style.backgroundColor = alpha(theme.palette.primary.main, 0.08);
    }
  };

  const handleDragLeave = (e: React.DragEvent<HTMLDivElement>) => {
    e.preventDefault();
    e.stopPropagation();
    if (dropAreaRef.current) {
      dropAreaRef.current.style.borderColor = alpha(theme.palette.primary.main, 0.3);
      dropAreaRef.current.style.backgroundColor = alpha(theme.palette.primary.main, 0.05);
    }
  };

  const handleDrop = (e: React.DragEvent<HTMLDivElement>) => {
```

```
    e.preventDefault();
    e.stopPropagation();
    if (dropAreaRef.current) {
      dropAreaRef.current.style.borderColor = alpha(theme.palette.primary.main, 0.3);
      dropAreaRef.current.style.backgroundColor = alpha(theme.palette.primary.main, 0.05);
    }

    if (e.dataTransfer.files && e.dataTransfer.files.length > 0) {
      const file = e.dataTransfer.files[0];
      if (!file.type.match('image.*')) {
        setError('Будь ласка, завантажте зображення');
        return;
      }

      const reader = new FileReader();
      reader.onload = (event) => {
        if (event.target) {
          setImage(event.target.result as string);
        }
      };
      reader.readAsDataURL(file);

      analyzeFood(file);
    }
  };

  const analyzeFood = async (file: File) => {
    setIsLoading(true);
    setError(null);
    setFoodData(null);

    const formData = new FormData();
    formData.append('foodImage', file);

    try {
      // Припускаємо, що сервер працює на порту 3001
      const response = await fetch('http://localhost:3001/analyze-food', {
        method: 'POST',
        body: formData,
      });

      if (!response.ok) {
        throw new Error(`Помилка сервера: ${response.status}`);
      }

      const data = await response.json();

      if (data.content && data.content[0] && data.content[0].text) {
        const claudeText = data.content[0].text;


        try {
```

```
      // Спочатку спробуємо знайти JSON у тексті
      const jsonMatch = claudeText.match(/```json([\s\S]*?)```/) ||
                claudeText.match(/{[\s\S]*?}/);


      if (jsonMatch) {
        const jsonString = jsonMatch[0].replace(/```json|```/g, '').trim();
        const parsedData = JSON.parse(jsonString);
        setFoodData(parsedData);
      } else {
        setError('Не вдалося розпізнати дані про нутрієнти');
      }
    } catch (e) {
      console.error('Помилка обробки JSON:', e);
      setError('Помилка обробки відповіді');
    }
  } else {
    setError('Отримано неправильний формат відповіді');
  }
 } catch (err: any) {
  setError(`Помилка: ${err.message}`);
 } finally {
  setIsLoading(false);
 }
};

const handleAddMeal = () => {
 if (foodData) {
  const portion = parseFloat(portionSize);
  const calories = Math.round((foodData.калорійність * portion) / 100);
  tracker.addMeal(foodData.назва, calories);


  // Створюємо елемент для анімації успіху
  const successElement = document.createElement('div');
  successElement.style.position = 'fixed';
  successElement.style.top = '20px';
  successElement.style.left = '50%';
  successElement.style.transform = 'translateX(-50%)';
  successElement.style.backgroundColor = theme.palette.success.main;
  successElement.style.color = 'white';
  successElement.style.padding = '12px 24px';
  successElement.style.borderRadius = '8px';
  successElement.style.zIndex = '9999';
  successElement.style.boxShadow = '0 4px 12px rgba(0, 0, 0, 0.15)';
  successElement.style.display = 'flex';
  successElement.style.alignItems = 'center';
  successElement.style.gap = '8px';
  successElement.style.fontFamily = '"Roboto", "Helvetica", "Arial", sans-serif';
  successElement.style.transition = 'all 0.3s ease';
  successElement.style.opacity = '0';
  successElement.style.transform = 'translateX(-50%) translateY(-20px)';
```

```
    const icon = document.createElement('span');
    icon.textContent = '✓';
    icon.style.fontWeight = 'bold';
    icon.style.fontSize = '20px';

    const text = document.createTextNode(`Додано страву "${foodData.назва}" (${calories} ккал)`);

    successElement.appendChild(icon);
    successElement.appendChild(text);
    document.body.appendChild(successElement);

    // Анімуємо появу
    setTimeout(() => {
      successElement.style.opacity = '1';
      successElement.style.transform = 'translateX(-50%) translateY(0)';
    }, 10);

    // Автоматично закриваємо через 3 секунди
    setTimeout(() => {
      successElement.style.opacity = '0';
      successElement.style.transform = 'translateX(-50%) translateY(-20px)';

      // Повністю видаляємо елемент після завершення анімації
      setTimeout(() => {
        document.body.removeChild(successElement);
      }, 300);
    }, 3000);

    // Очищаємо дані після додавання
    setFoodData(null);
    setImage(null);
    setPortionSize('100');
  }
};

const handleRetry = () => {
  setImage(null);
  setError(null);
  setFoodData(null);
};

const calculateNutrientForPortion = (value: number): number => {
  const portion = parseFloat(portionSize);
  return Math.round((value * portion) / 100 * 10) / 10;
};
```

```
const getNutrientIcon = (nutrient: string) => {
  switch(nutrient.toLowerCase()) {
    case 'калорійність':
      return <LocalDiningIcon />;
    case 'білки':
      return <EggIcon />;
    case 'жири':
      return <OilBarrelIcon />;
    case 'вуглеводи':
      return <GrainIcon />;
    case 'клітковина':
      return <BlenderIcon />;
    default:
      return <VitaminIcon />;
  }
};

return (
  <Grid container spacing={3}>
    {!image && !isLoading && !foodData && (
      <Grid item xs={12}>
        <Box
          ref={dropAreaRef}
          sx={{
            borderRadius: 2,
            borderStyle: 'dashed',
            borderWidth: 2,
            borderColor: alpha(theme.palette.primary.main, 0.3),
            p: 6,
            display: 'flex',
            flexDirection: 'column',
            alignItems: 'center',
            backgroundColor: alpha(theme.palette.primary.main, 0.05),
            transition: 'all 0.3s ease',
            '&:hover': {
              borderColor: theme.palette.primary.main,
              backgroundColor: alpha(theme.palette.primary.main, 0.08),
            },
            cursor: 'pointer'
          }}
          onClick={() => fileInputRef.current?.click()}
          onDragOver={handleDragOver}
          onDragLeave={handleDragLeave}
          onDrop={handleDrop}
        >
          <ImageIcon sx={{ fontSize: 80, color: alpha(theme.palette.primary.main, 0.6), mb: 3 }} />
          <Typography variant="h5" sx={{ mb: 1, fontWeight: 500 }}>Завантажте фото їжі</Typography>
          <Typography variant="body1" sx={{ mb: 4, textAlign: 'center', color: 'text.secondary' }}>
            Перетягніть зображення сюди або натисніть кнопку нижче
          </Typography>
```

```
      <Button
        variant="contained"
        startIcon={<PhotoCameraIcon />}
        disabled={isLoading}
        size="large"
        sx={{
          px: 4,
          py: 1.5,
          boxShadow: '0 4px 12px rgba(0, 0, 0, 0.1)',
          '&:hover': {
            boxShadow: '0 6px 16px rgba(0, 0, 0, 0.2)',
          }
        }}
      >
        Вибрати зображення
      </Button>


      <input
        type="file"
        accept="image/*"
        style={{ display: 'none' }}
        ref={fileInputRef}
        onChange={handleImageUpload}
      />
    </Box>
  </Grid>
)}


{image && !isLoading && !foodData && !error && (
  <Grid item xs={12}>
    <Card>
      <CardContent sx={{ textAlign: 'center', p: 3 }}>
        <Typography variant="h6" sx={{ mb: 3, display: 'flex', alignItems: 'center', justifyContent: 'center' }}>
          <PhotoCameraIcon sx={{ mr: 1 }} />
          Фото для аналізу
        </Typography>
        <Box sx={{ textAlign: 'center', mb: 3 }}>
          <img
            src={image}
            alt="Завантажене фото"
            style={{
              maxWidth: '100%',
              maxHeight: '400px',
              borderRadius: theme.shape.borderRadius,
              boxShadow: '0 4px 12px rgba(0, 0, 0, 0.15)'
            }}
          />
        </Box>
        <Typography variant="body2" color="text.secondary" sx={{ mb: 3 }}>
```

```
          Очікуємо аналізу... Зображення готове до обробки
        </Typography>
      </CardContent>
    </Card>
  </Grid>
)}


{isLoading && (
  <Grid item xs={12}>
    <Card>
      <CardContent sx={{ display: 'flex', flexDirection: 'column', alignItems: 'center', py: 5 }}>
        <CircularProgress size={60} thickness={4} sx={{ mb: 3 }} />
        <Typography variant="h5" sx={{ mb: 2 }}>Аналізуємо зображення...</Typography>
        <Typography variant="body1" color="text.secondary" sx={{ mb: 1 }}>
          Розпізнаємо страву та її нутрієнтний склад
        </Typography>
        <Typography variant="body2" color="text.secondary">
          Це може зайняти кілька секунд
        </Typography>
      </CardContent>
    </Card>
  </Grid>
)}


{error && (
  <Grid item xs={12}>
    <Alert
      severity="error"
      variant="filled"
      sx={{
        borderRadius: 2,
        '& .MuiAlert-icon': { fontSize: '1.5rem' },
        mb: 2
      }}
    >
      {error}
    </Alert>
    <Box sx={{ display: 'flex', justifyContent: 'center' }}>
      <Button
        variant="outlined"
        color="primary"
        onClick={handleRetry}
        startIcon={<PhotoCameraIcon />}
      >
        Спробувати інше зображення
      </Button>
    </Box>
  </Grid>
)}
```

```
{foodData && (
 <Grid item xs={12}>
  <Card sx={{ overflow: 'hidden' }}>
   <Box sx={{ bgcolor: 'primary.main', color: 'white', p: 3 }}>
    <Typography variant="h5" sx={{ mb: 1, display: 'flex', alignItems: 'center' }}>
     <FastfoodIcon sx={{ mr: 1 }} />
     {foodData.назва}
    </Typography>
    <Typography variant="body2" sx={{ opacity: 0.8 }}>
     Результати аналізу харчової цінності
    </Typography>
   </Box>


   <CardContent>
    <Grid container spacing={3} sx={{ mb: 4, mt: 1 }}>
     <Grid item xs={12} sm={6}>
      <Paper
       elevation={0}
       sx={{
        p: 2,
        bgcolor: alpha(theme.palette.primary.main, 0.05),
        borderRadius: 2
       }}
      >
       <Typography variant="subtitle2" sx={{ color: 'text.secondary', mb: 1 }}>
        Розмір порції
       </Typography>
       <Box sx={{ display: 'flex', alignItems: 'center', gap: 2 }}>
        <TextField
         label="Bara"
         type="number"
         value={portionSize}
         onChange={(e) => setPortionSize(e.target.value)}
         fullWidth
         variant="outlined"
         size="small"
         InputProps={{
          startAdornment: <ScaleIcon sx={{ mr: 1, color: 'primary.main' }} />,
          endAdornment: <Typography variant="body2" sx={{ ml: 1 }}>г</Typography>
         }}
        />
        <Button
         variant="contained"
         color="primary"
         startIcon={<AddIcon />}
         onClick={handleAddMeal}
         sx={{
          height: 40,
          whiteSpace: 'nowrap',
          boxShadow: '0 4px 12px rgba(0, 0, 0, 0.1)',
         }}
```

```
        >
          Додати
        </Button>
      </Box>
    </Paper>
  </Grid>


  <Grid item xs={12} sm={6}>
    <Paper
      elevation={0}
      sx={{
        p: 2,
        bgcolor: alpha(theme.palette.primary.main, 0.05),
        borderRadius: 2,
        height: '100%',
        display: 'flex',
        alignItems: 'center'
      }}
    >
      <Grid container spacing={2}>
        <Grid item xs={6}>
          <Typography variant="subtitle2" sx={{ color: 'text.secondary', mb: 1 }}>
            Калорійність:
          </Typography>
          <Typography variant="h4" color="primary" sx={{ fontWeight: 'bold' }}>
            {calculateNutrientForPortion(foodData.калорійність)}
          </Typography>
          <Typography variant="body2" color="text.secondary">
            ккал
          </Typography>
        </Grid>
        <Grid item xs={6}>
          <Typography variant="subtitle2" sx={{ color: 'text.secondary', mb: 1 }}>
            На 100г:
          </Typography>
          <Typography variant="h5" sx={{ fontWeight: 'medium' }}>
            {foodData.калорійність}
          </Typography>
          <Typography variant="body2" color="text.secondary">
            ккал
          </Typography>
        </Grid>
      </Grid>
    </Paper>
  </Grid>
</Grid>


<Typography variant="h6" sx={{ mb: 2, display: 'flex', alignItems: 'center' }}>
  <InfoIcon sx={{ mr: 1, color: 'primary.main' }} />
  Нутрієнтний склад
```

```
</Typography>


<TableContainer component={Paper} variant="outlined" sx={{ mb: 3, borderRadius: 2 }}>
 <Table>
  <TableHead>
   <TableRow sx={{ bgcolor: alpha(theme.palette.primary.main, 0.05) }}>
    <TableCell sx={{ fontWeight: 'bold' }}>Нутрієнт</TableCell>
    <TableCell align="right" sx={{ fontWeight: 'bold' }}>На 100г</TableCell>
    <TableCell align="right" sx={{ fontWeight: 'bold' }}>У порції ({portionSize}г)</TableCell>
   </TableRow>
  </TableHead>
  <TableBody>
   {Object.entries(foodData).map(([key, value]) => {
    // Показуємо тільки числові значення, крім назви
    if (key !== 'назва' && typeof value === 'number') {
     return (
      <TableRow
       key={key}
       sx={{
        '&:hover': {
         backgroundColor: alpha(theme.palette.primary.main, 0.04)
        }
       }}
      >
       <TableCell sx={{ display: 'flex', alignItems: 'center' }}>
        <Box sx={{
         mr: 1.5,
         color: 'primary.main',
         display: 'flex',
         alignItems: 'center'
        }}>
         {getNutrientIcon(key)}
        </Box>
        {key}
       </TableCell>
       <TableCell align="right">{value} {key === 'калорійність' ? 'ккал' : 'г'}</TableCell>
       <TableCell align="right" sx={{ fontWeight: 'medium' }}>
        {calculateNutrientForPortion(value)} {key === 'калорійність' ? 'ккал' : 'г'}
       </TableCell>
      </TableRow>
     );
    }
    return null;
   })}
  </TableBody>
 </Table>
</TableContainer>


{(foodData.вітаміни || foodData.мінерали) && (
 <Box>
```

```
<Divider sx={{ mb: 3 }} />
<Grid container spacing={3}>
 {foodData.вітаміни && (
  <Grid item xs={12} md={6}>
   <Typography variant="subtitle1" sx={{ mb: 1.5, display: 'flex', alignItems: 'center' }}>
    <VitaminIcon sx={{ mr: 1, color: 'primary.main' }} />
    Вітаміни
   </Typography>
   <Box sx={{ display: 'flex', flexWrap: 'wrap', gap: 1 }}>
    {Array.isArray(foodData.вітаміни) ?
     foodData.вітаміни.map((vitamin, idx) => (
      <Chip
       key={idx}
       label={vitamin}
       size="small"
       sx={{
        bgcolor: alpha(theme.palette.primary.main, 0.1),
        color: 'primary.main'
       }}
      />
     )) :
     <Typography variant="body2">{foodData.вітаміни}</Typography>
    }
   </Box>
  </Grid>
 )}


 {foodData.мінерали && (
  <Grid item xs={12} md={6}>
   <Typography variant="subtitle1" sx={{ mb: 1.5, display: 'flex', alignItems: 'center' }}>
    <VitaminIcon sx={{ mr: 1, color: 'secondary.main' }} />
    Мінерали
   </Typography>
   <Box sx={{ display: 'flex', flexWrap: 'wrap', gap: 1 }}>
    {Array.isArray(foodData.мінерали) ?
     foodData.мінерали.map((mineral, idx) => (
      <Chip
       key={idx}
       label={mineral}
       size="small"
       sx={{
        bgcolor: alpha(theme.palette.secondary.main, 0.1),
        color: 'secondary.main'
       }}
      />
     )) :
     <Typography variant="body2">{foodData.мінерали}</Typography>
    }
   </Box>
  </Grid>
 )}
```

```
            </Grid>
          </Box>
        )}

        <Box sx={{ display: 'flex', justifyContent: 'space-between', mt: 4 }}>
          <Button
            variant="outlined"
            color="primary"
            startIcon={<PhotoCameraIcon />}
            onClick={handleRetry}
          >
            Нове фото
          </Button>
          <Button
            variant="contained"
            color="primary"
            startIcon={<AddIcon />}
            onClick={handleAddMeal}
            sx={{
              boxShadow: '0 4px 12px rgba(0, 0, 0, 0.1)',
              '&:hover': {
                boxShadow: '0 6px 16px rgba(0, 0, 0, 0.2)',
              }
            }}
          >
            Додати до раціону
          </Button>
        </Box>
      </CardContent>
    </Card>
  </Grid>
 )}
</Grid>
);
};

export default FoodAnalyzer;
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ КОНТРОЛЮ БІОЛОГІЧНИХ ПОКАЗНИКІВ ЛЮДИНИ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

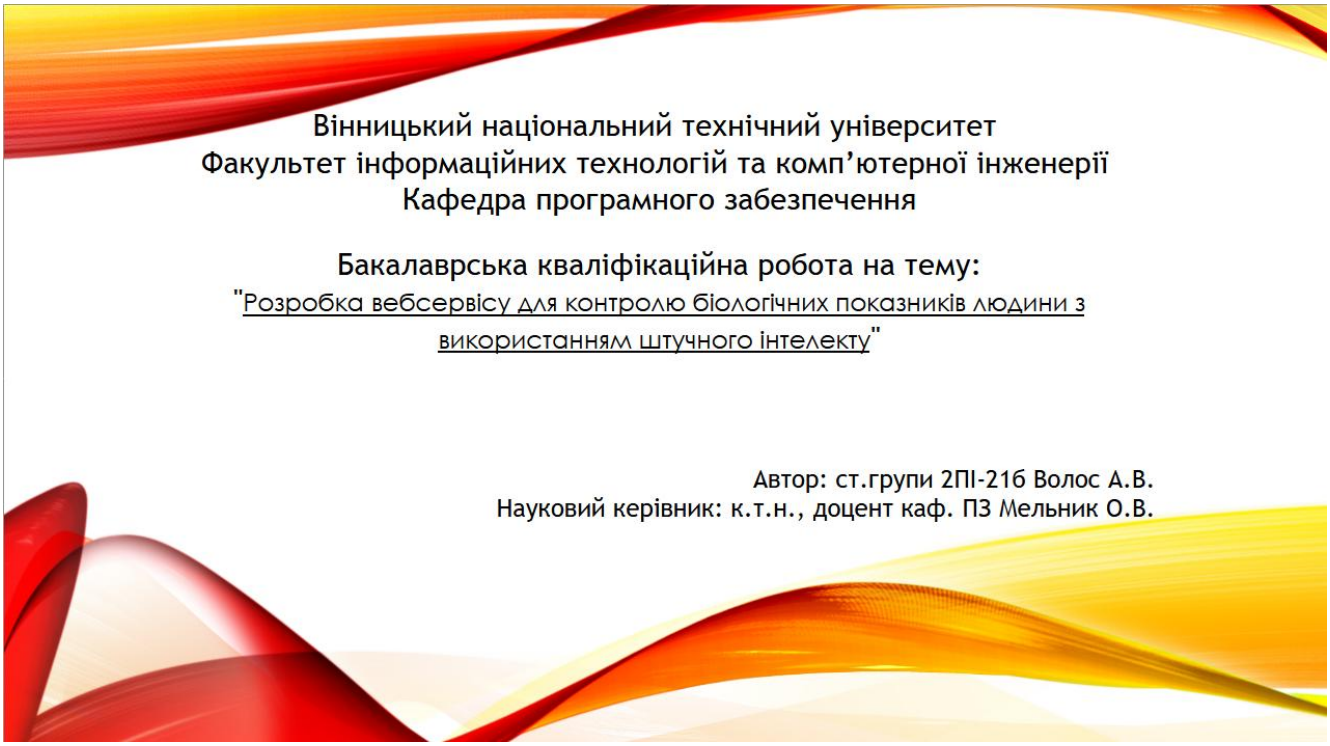


Рисунок Г.1 – Титульний слайд

Моніторинг здоров‘я, або моніторинг mHealth, популярний у цифровому секторі охорони здоров’я завдяки використанню портативних пристроїв, таких як смартфони, планшети та портативні комп‘ютери, для збору та передачі інформації про здоров‘я з метою керування здоров‘ям користувача.

Прагнення сучасної людини до здорового способу життя призвело до популяризації додатків для дослідження біологічних показників. Вони полегшують відстеження калорій, активності, сну та інших важливих показників здоров‘я. Однак ці застосунки характеризуються неточністю даних, бездіяльністю користувачів та інформаційною безпекою. У контексті технологічного прогресу вдосконалення цих ініціатив є пріоритетом для підвищення їх ефективності та цінності. Трекери здоров'я пропонують користувачам інструменти для спостереження за своїм фізичним здоров'ям на особистому рівні.

Багато додатків пропонують неперсоналізовані рекомендації, що не враховують індивідуальних потреб та стану здоров'я користувача. Це може призводити до низької ефективності програм або навіть до шкоди.

Перевантаження даними без зрозумілих візуалізацій ускладнює інтерпретацію інформації та її корисність. Проблеми з мотивацією і тривалістю використання пов'язані з недостатньою гейміфікацією та відсутністю індивідуальних досягнень. Надійність і точність даних є важливими аспектами, оскільки неточні дані можуть негативно вплинути на прийняття рішень користувачами. Останнім, але не менш важливим аспектом є забезпечення конфіденційності та безпеки особистих даних, що є важливим для довіри користувачів до таких додатків.

Отже, актуальною є розробка застосунку для моніторингу біологічних показників людини.

Рисунок Г.2 — Актуальність теми

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Метою роботи є підвищення рівня контролю біологічних показників людини шляхом розробки спеціалізованої системи, що дозволяє відслідковування та контроль біологічних показників людини.

Об'єкт дослідження – процес розробки системи для контролю біологічних показників людини з використанням штучного інтелекту.

Предмет дослідження – методи та засоби розробки системи для контролю біологічних показників людини з використанням штучного інтелекту.

Рисунок Г.3 — Мета, об'єкт та предмет дослідження

ЗАВДАННЯ ДОСЛІДЖЕННЯ

– розробити архітектуру застосунку для контролю біологічних показників людини;
– розробити інтерфейс користувача застосунку;
– розробити алгоритми роботи застосунку;
– розробити функціонал застосунку;
– провести тестування розробленого застосунку.

Рисунок Г.4 — Завдання дослідження

НОВИЗНА

1. Подальшого розвитку отримав метод автоматичного аналізу харчування на основі комп'ютерного зору, який, на відміну від відомих, дозволяє провести аналіз нутрієнтної цінності раціону харчування по його фото.

2. Подальшого розвитку отримав метод персоналізованих адаптивних рекомендацій, який на відміну від відомих, надає персоналізовані рекомендації, які залежать від харчування користувача, його фізичної активності, бажаної ваги та мети використання застосунку.

Рисунок Г.5 — Новизна

ПРАКТИЧНА ЦІННІСТЬ

Практична цінність одержаних результатів полягає у можливості використання розробленої системи для контролю біологічних показників людини, аналізу харчування, відслідковування ваги.

Рисунок Г.6 — Практична цінність

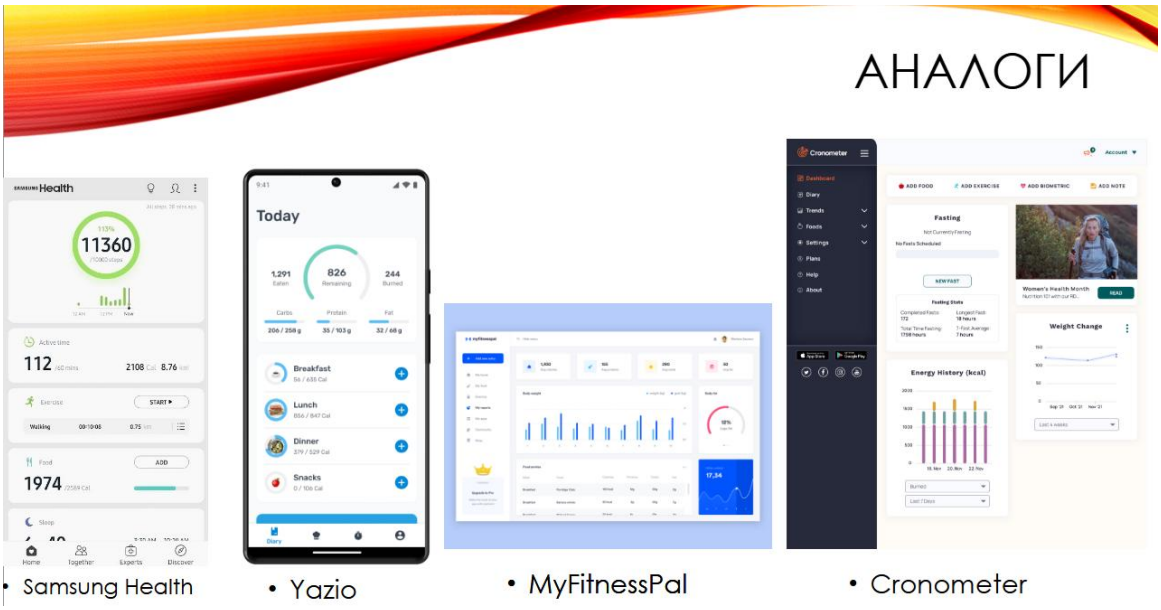


Рисунок Г.7 — Аналоги

ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ

Характеристика	Samsung Health	MyFitnessPal	Yazio	Cronometer	Власна розробка
Автоматичне розпізнавання їжі за фотографією	-	+	-	-	+
Динамічний перерахунок нутрієнтів за вагою порції	-	-	+	+	+
Персоналізовані рекомендації на основі даних харчування	+	+	+	+	+
Відстеження макронутрієнтів	+	+	+	+	+
Відстеження мікронутрієнтів	-	-	-	+	+
Відстеження фізичної активності	+	+	+	+	+
Деталізований аналіз складу страви	-	-	-	+	+
Сумарний бал	3	4	4	6	7

Рисунок Г.8 — Порівняльний аналіз аналогів

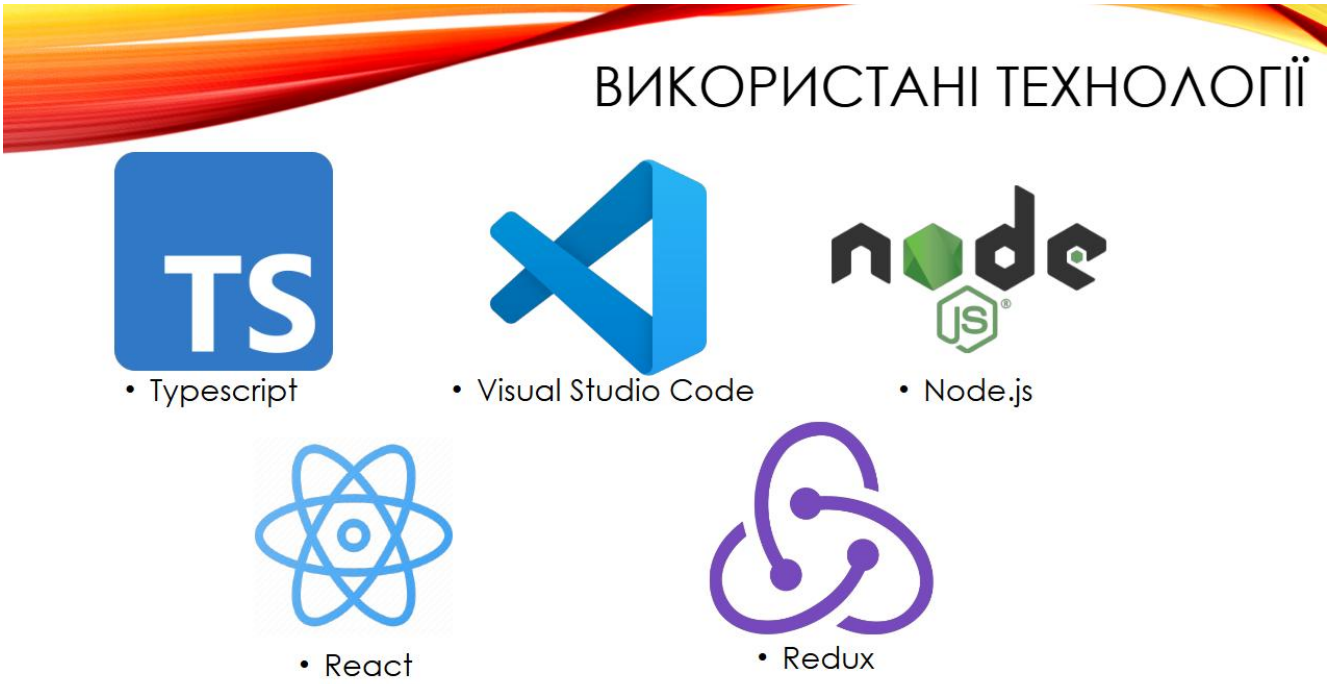
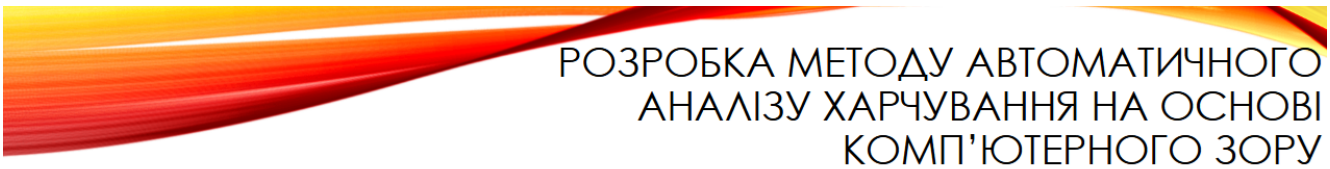


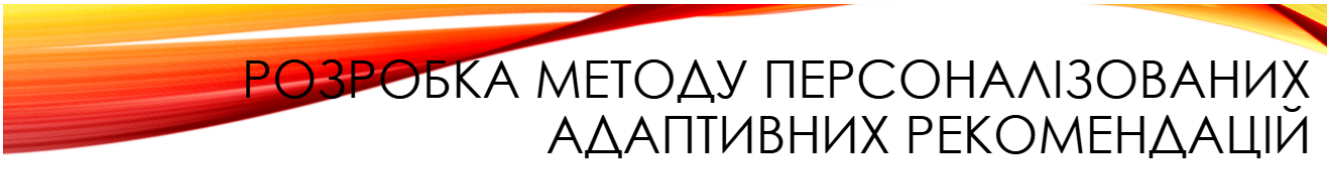
Рисунок Г.9 — Використані технології



1 . Користувач робить фотографію страви або завантажує існуюче зображення через інтерфейс застосунку.
2 . Система конвертує зображення у формат, придатний для обробки (base64), та передає його до модуля комп'ютерного зору.
3 . Модуль комп'ютерного зору, інтегрований з великою мовною моделлю (Claude Vision API), аналізує візуальні характеристики страви, включаючи колір, текстуру, форму компонентів та їх розташування.
4 . Система ідентифікує тип страви та визначає її стандартний нутрієнтний склад на 100 грамів, включаючи калорійність, макронутрієнти (білки, жири, вуглеводи), мікронутрієнти (вітаміни, мінерали) та інші показники (клітковина, цукри, насичені жири).
5 . Отримана інформація структурується у стандартизованому форматі JSON для подальшої обробки та зберігання.
6 . Система запитує користувача про фактичну вагу спожитої порції або пропонує оцінити її на основі візуальних характеристик зображення та типових розмірів порцій для даної страви.
7 . Алгоритм динамічного перерахунку застосовує математичну модель пропорційного масштабування до кожного нутрієнта за формулою:
нутрієнт_факт = нутрієнт_стандарт × вага_фактична / 100.
8 . Система зберігає як початкові дані про нутрієнти (на 100 г), так і розраховані значення для фактичної порції, забезпечуючи можливість подальшого перерахунку без втрати точності.
9 . На основі актуальних даних про споживання формується візуальне відображення нутрієнтного складу з урахуванням рекомендованих добових норм споживання.
10 . Інформація про розпізнану страву та її нутрієнтний склад зберігається в базі даних з прив'язкою до часової мітки, що дозволяє формувати хронологічні звіти про харчування.
11 . При кожній зміні введеної користувачем ваги порції система оперативно перераховує значення всіх нутрієнтів, зберігаючи високу точність даних.
12 . На основі накопичених даних система може формувати аналітичні звіти про відповідність фактичного споживання нутрієнтів індивідуальним потребам користувача.

Рисунок Г.10 — Розробка методу автоматичного аналізу харчування на основі комп’ютерного зору

Рисунок Г.11 — Алгоритми методу автоматичного аналізу харчування на основі комп'ютерного зору



1. Система накопичує базу даних харчової поведінки користувача, створюючи детальний профіль користувача.
2. На основі антропометричних даних (вік, стать, вага, зріст) та рівня активності розраховуються індивідуальні потреби в калоріях, ключових нутрієнтах, та фізичних вправах.
3. Користувач визначає свої цілі (зниження ваги, набір м'язової маси, підтримка здоров'я) та вказує медичні обмеження (алергії, хронічні захворювання).
4. Алгоритм аналізує розбіжності між фактичним споживанням нутрієнтів та індивідуальними потребами, ідентифікуючи дефіцити та надлишки конкретних поживних речовин.
5. На основі даних профілю система ідентифікує харчові переваги користувача, визначаючи страви, які споживаються регулярно.
6. Формується набір рекомендованих продуктів та страв, що компенсують виявлені дефіцити нутрієнтів, відповідають цілям та обмеженням користувача, враховуючи його харчові вподобання.
7. Система використовує механізми прогресивного навчання для адаптації рекомендацій на основі зворотного зв'язку від користувача та даних про фактичне споживання рекомендованих страв.
8. Через інтерфейс застосунку користувач отримує персоналізовані рекомендації щодо оптимальних страв, альтернативних продуктів та пропорцій прийомів їжі і фізичних вправ.
9. Періодичне порівняння цільових показників з фактичними результатами дозволяє системі оптимізувати стратегію рекомендацій для досягнення найкращого результату.

Рисунок Г.12 — Розробка методу персоналізованих адаптивних рекомендацій

Рисунок Г.13 — Алгоритм методу персоналізованих адаптивних рекомендацій

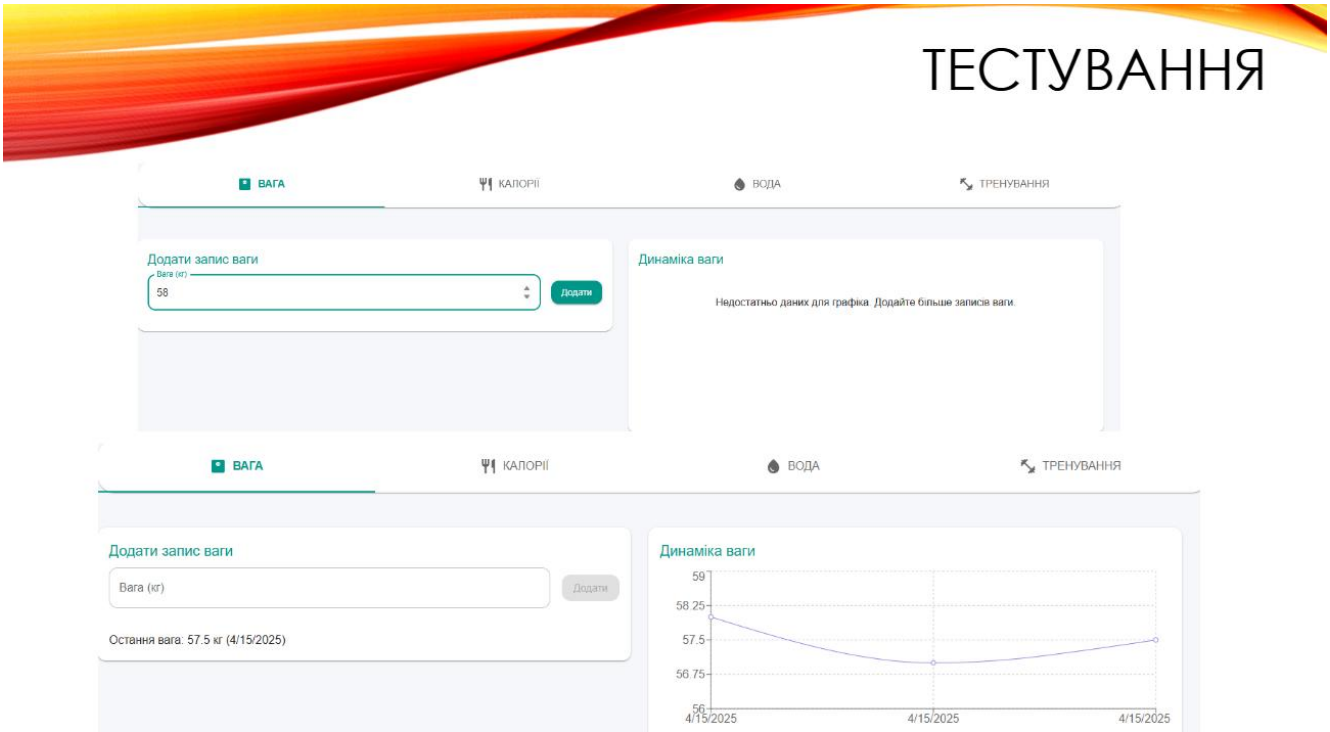


Рисунок Г.14 — Тестування функціоналу відслідковування ваги

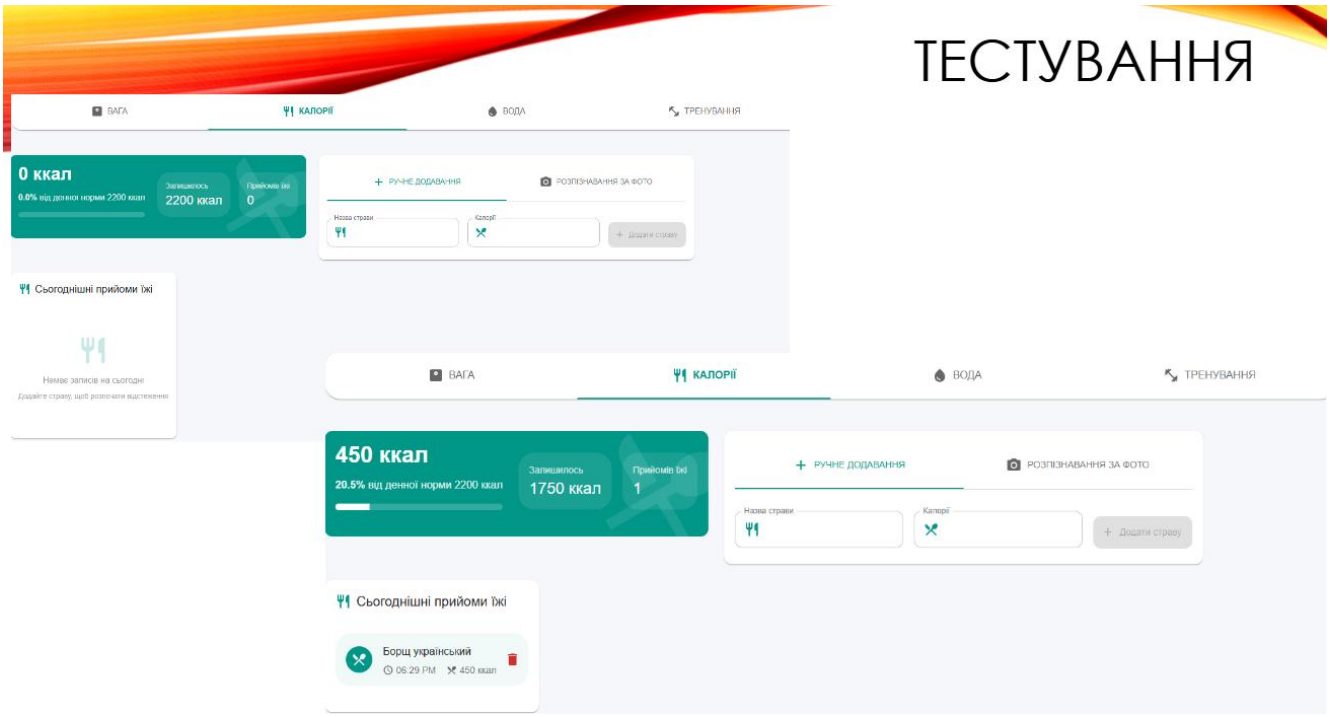


Рисунок Г.15 — Тестування функцоіналу для відстежування спожитих нутрієнтів

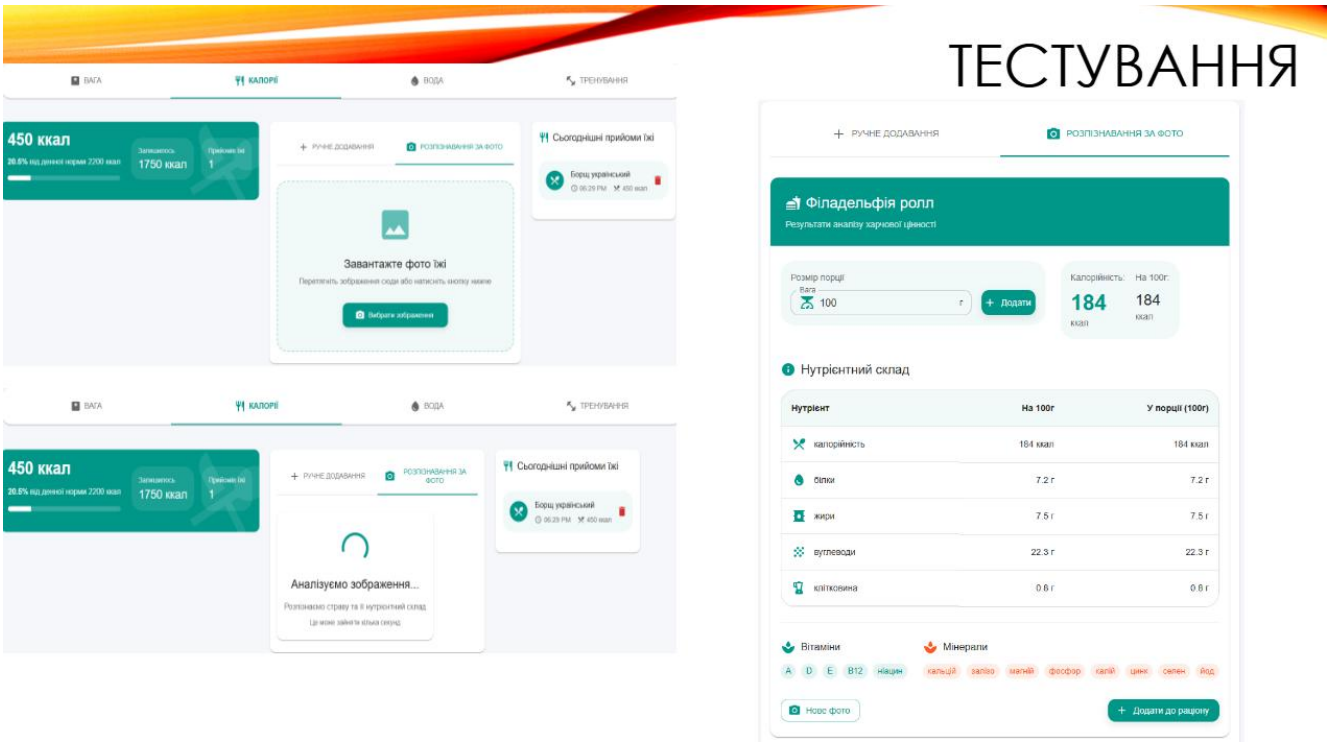


Рисунок Г.16 — Тестування функціоналу для аналізу спожитої страви

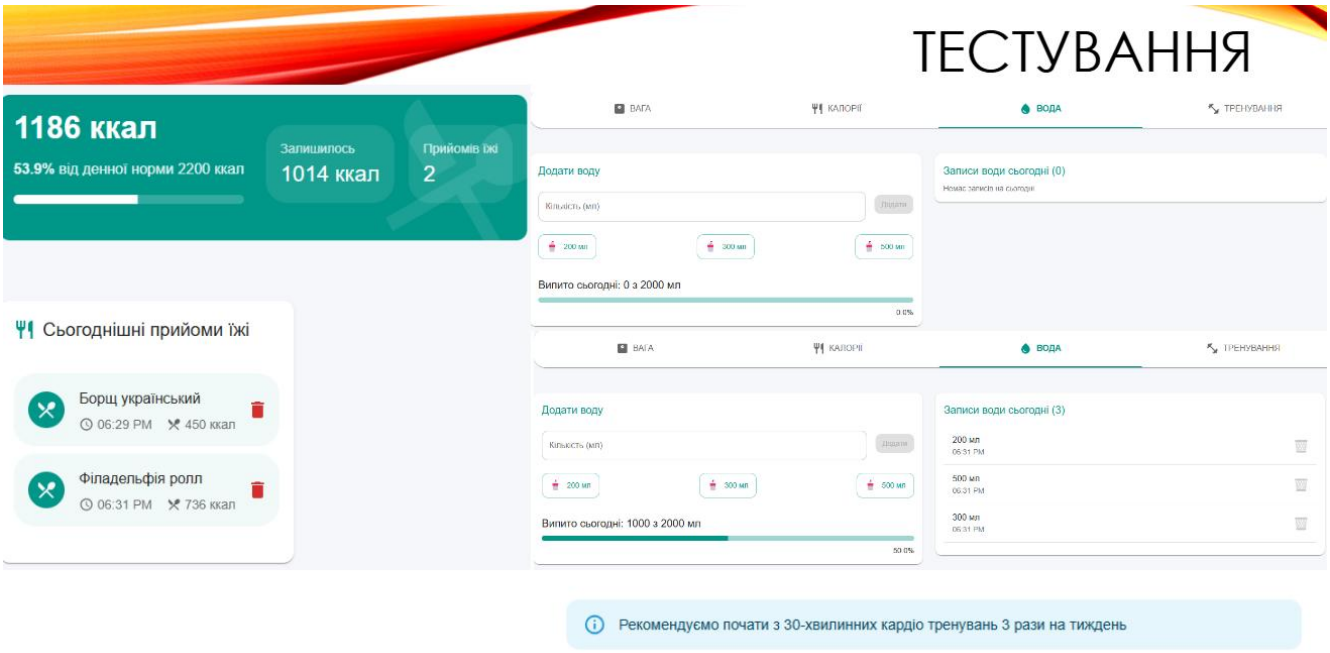


Рисунок Г.17 — Тестування функціоналу відстеження спожитої води

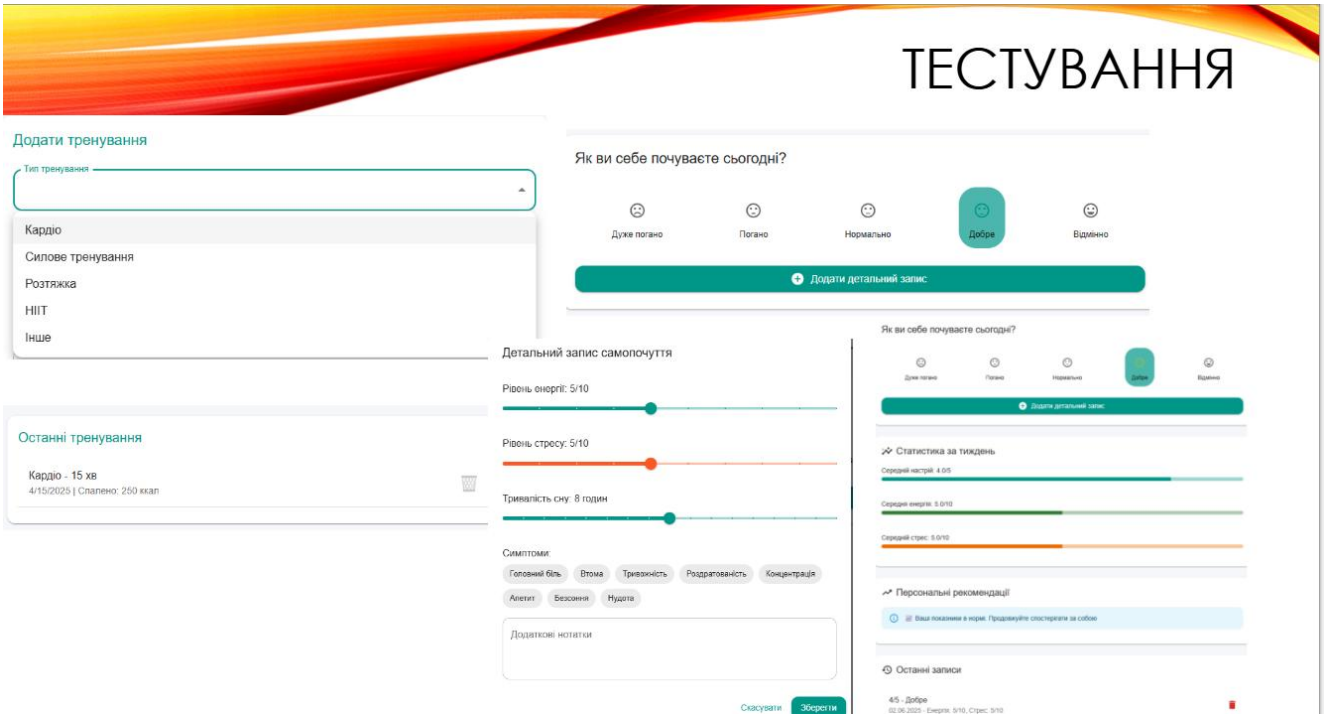


Рисунок Г.18 — Тестування функціоналу відстеження виконаних вправ та якості сну

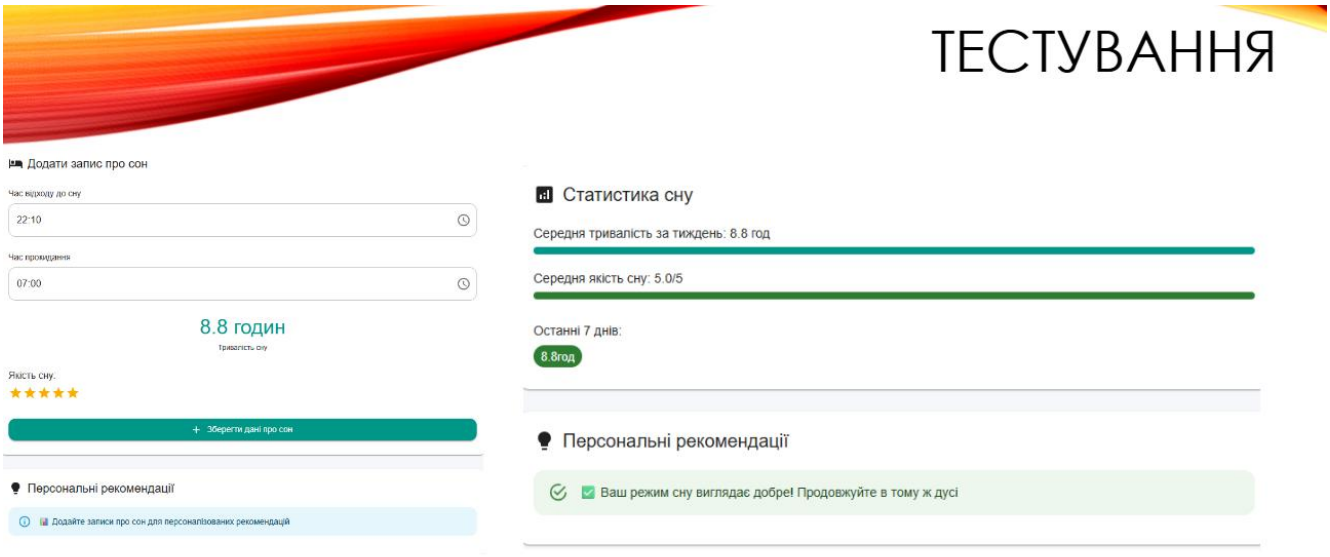


Рисунок Г.19 — Тестування функціоналу додавання інформації про сон

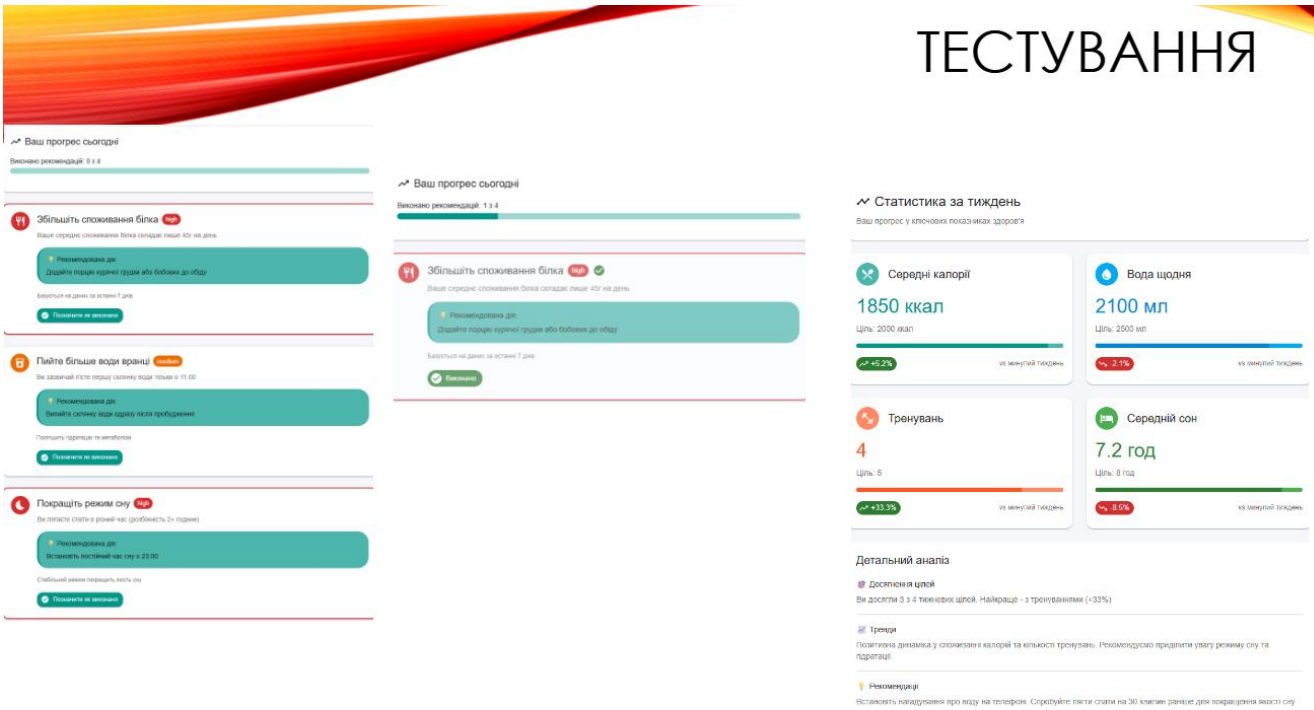


Рисунок Г.20 — Тестування функціоналу статистики

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи, було розроблено вебсервіс для контролю біологічних показників людини з використанням штучного інтелекту. Розроблений сервіс дозволяє відстежувати зміну ваги користувача, проводити аналіз спожитої їжі, випитої води, отримувати рекомендації щодо тренування та харчування.

Було проведено аналіз сучасного стану застосунків для контролю біологічних показників людини, доведено актуальність програмної розробки в цьому напрямку, проведено порівняльний аналіз аналогів.

Для розробки було обрано мову програмування TypeScript та середовище розробки Visual Studio Code, Claude Vision API для розпізнавання їжі за фото, Node.js для серверної частини застосунку, React.js для інтерфейсу користувача. Бакалаврську кваліфікаційну роботу реалізовано згідно методичних вказівок.

У першому розділі було проведено аналіз сучасного стану застосунків для контролю біологічних показників людини, доведено актуальність програмної розробки в цьому напрямку, проведено порівняльний аналіз аналогів: Samsung Health, MyFitnessPal, Yazio, Cronometer, продемонстровано переваги власної розробки перед аналогами, проведено аналіз методів розв'язання задачі, поставлено задачі дослідження на розробку вебзастосунку для контролю біологічних показників людини з використанням штучного інтелекту.

Рисунок Г.21 — Висновки (частина 1)

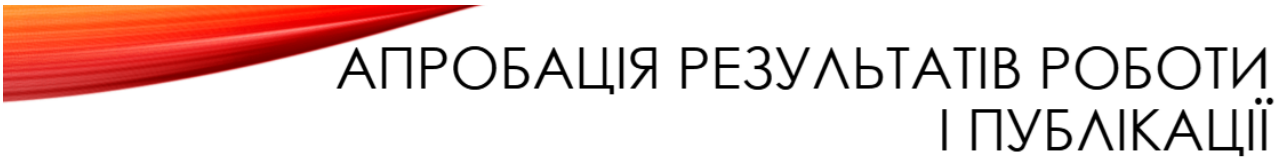
ВИСНОВКИ

У другому розділі було проведено аналіз інформаційного забезпечення вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту, визначено дані, які збираються, обробляються, генеруються при роботі вебсервісу. Розроблено модель системи у три шари – клієнтська частина, серверна та мережева. Описано компоненти, що використовуються у роботі вебсервісу. Розроблено архітектуру вебсервісу, побудовано деревовидну структуру, що містить усі класи, методи, таблиці для роботи вебсервісу. Розроблено методи автоматичного аналізу харчування на основі комп'ютерного зору та персоналізованих адаптивних рекомендацій.

У третьому розділі було проведено аналіз мов програмування для розробки вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту, було обрано мову програмування TypeScript. Розроблено базу даних вебсервісу, сформовано схему бази даних, описано таблиці розробленої бази даних, які дозволяють зберігати, обробляти та витягувати дані згідно потреб користувача та функціоналу роботи системи. Розроблено інтерфейс користувача, наведено структуру інтерфейсу усіх вікон розробленого вебсервісу. Розроблений інтерфейс дозволяє додавати записи про спожиту їжу, воду, вести облік зміни ваги та отримувати інформацію про нутрієнтну цінність спожитої їжі.

У четвертому розділі було проведено аналіз методів тестування вебсервісу для контролю біологічних показників людини з використанням штучного інтелекту. Проведено тестування розробленого сервісу. Було продемонстровано його можливості, наведено приклад використання, продемонстровано, що розроблений застосунок виконує поставлені на початку розробки завдання.

Рисунок Г.22 — Висновки (частина 2)



Апробація результатів роботи. Описані у роботі положення доповідались на конференції «IV Всеукраїнська науково-технічна конференції молодих вчених, аспірантів і студентів».
Публікації. Результати роботи були опубліковані в матеріалах конференції «IV Всеукраїнська науково-технічна конференції молодих вчених, аспірантів і студентів».

Рисунок Г.23 — Апробація результатів роботи і публікації